U.S. DEPARTMENT OF COMMERCE
Patent and Trademark Office

DOCUMENT RETRIEVAL REQUEST FORM

Please include RightFax Number to expedite return of documents

Requester's Name: TAN MAI	Case Serial Number: 09/977,985		Art Unit/Org.: 2124
Phone: 703-305-9761	**RightFax:	Building: CPK 2	Room Number: 5405
Class/Sub-Class: 708/250			
Date of Request: 4/21/04		Date Needed By:	

Paste or add text of citation or bibliography: **Paste Citation** Only one request per form. Original copy only. ☐

Author/Editor:	S. Wolfram
Journal/Book Title:	Advances in Applied Mathematics
Article Title:	Random sequence generation by cellular automata
Volume Number:	7 Report Number: Pages: 123-169
Issue Number:	Series Number: Year of Publication: Jun 1986
Publisher:	
Remarks:	271

Staff Use Only

Monthly Accession Number: 492864

Library Action	PTO 1st	PTO 2nd	LC 1st	LC 2nd	NAL 1st	NAL 2nd	NIH 1st	NIH 2nd	NLM 1st	NLM 2nd	NIST 1st	NIST 2nd	Other 1st	Other 2nd
Local Attempts	✓													
Date	4/22													
Initials	SG													
Results	Comp													
Examiner Called														
Page Count														
Money Spent														

	Source	Date
Ordered From:		

Remarks/Comments
1st and 2nd denotes time taken to a library

O/N – Under NLM means Overnight

Comments:

Random Sequence Generation by Cellular Automata (1986)

Reference: S. Wolfram: Advances in Applied Mathematics, 7 (June 1986) 123-169

Abstract

A 1-dimensional cellular automaton which generates random sequences is discussed. Each site in the cellular automaton has value 0 or 1, and is updated in parallel according to the rule $a_i' = a_{i-1}$ XOR $(a_i$ OR $a_{i+1})$ $(a_i' = (a_{i-1} + a_i + a_{i+1} + a_i a_{i+1}) \bmod 2)$. Despite the simplicity of this rule, the time sequences of site values that it yields seem to be completely random. These sequences are analysed by a variety of empirical, combinatorial, statistical, dynamical systems theory and computation theory methods. An efficient random sequence generator based on them is suggested.

Next section

1. Random Sequence Generation

Sequences that seem random are needed for a wide variety of purposes. They are used for unbiased sampling in the Monte Carlo method, and to imitate stochastic natural processes. They are used in implementing randomized algorithms which require arbitrary choices. And their unpredictability is used in games of chance, and potentially in data encryption.

To generate a random sequence on a digital computer, one starts with a fixed length seed, then iteratively applies some transformation to it, progressively extracting as long as possible a random sequence (e.g., [1]). In general one considers a sequence ``random'' if no patterns can be recognized in it, no predictions can be made about it, and no simple description of it can be found (e.g., [2]). But if in fact the sequence can be generated by iteration of a definite transformation, then a simple description of it certainly does exist. (1) The sequence can nevertheless seem random if no computations done on it reveal this simple description. The original seed must be transformed in such a complicated way that the computations cannot recover it.

The degree of randomness of a sequence can be defined in terms of the classes of computations which cannot discern patterns in it. A sequence is ``random enough'' for application in a particular system if the computations that the system effectively performs are not sophisticated enough to be able to find patterns in the sequence. So, for example, a sequence might be random enough for Monte Carlo integration if the values it yields are distributed sufficiently uniformly. The existence say of particular correlations in the sequence might not be discerned in this calculation. Whenever a computation that uses a random sequence takes a bounded time, there is a limit to the degree of randomness that the sequence need have. Statistical tests of randomness emulate various simple computations encountered in practice, and check that statistical properties of the sequence agree with those predicted if every element occurred purely according to probabilities. It would be better if one could show in general that patterns could not be recognized in certain sequences by any computation whatsoever that, for example, takes less than a certain time. No such results can yet be proved, so one must for now rely on more circumstantial evidence for adequate degrees of randomness.

The fact that acceptably random sequences can indeed be generated efficiently by digital computers is a consequence of the fact that quite simple transformations, when iterated, can yield extremely complicated behaviour. Simple computations are able to produce sequences whose origins can apparently be deduced only by much more complex computations.

Most current practical random sequence generation computer programs are based on linear congruence relations (of the form $x' = ax + b \bmod n$) (e.g., [1]), or linear feedback shift registers [4] (analogous to the linear cellular automata discussed below). The linearity and simplicity of these systems has made complete algebraic analyses possible and has allowed certain randomness properties to be proved [1, 4]. But it also leads to efficient algebraic algorithms for predicting the sequences (or deducing their seeds), and limits their degree of randomness.

An efficient random sequence generator should produce a sequence of length L in a time at most polynomial in L (and linear on most kinds of computers). It is always possible to deduce the seed (say of length s) for such a sequence by an exhaustive search which takes a time at most $O(2^s)$. But if in fact such an exponentially long computation were needed to find any pattern in the sequence, then the sequence would be random enough for almost any practical application (so long as it involved less than exponential time computations).

No such lower bounds on computational complexity are yet known. It is however often possible to show that one problem is computationally equivalent to a large class of others. So, for example, one could potentially show that the problem of deducing the seed for certain sequences was NP-complete [5]: special instances of the problem would then correspond to arbitrary problems in the class NP, and the problem would in general be as difficult as any in NP. (One should also show some form of uniform reducibility to ensure that the problem is difficult almost always, as well as in the worst case.) The class NP (nondeterministic polynomial time) includes many well-studied problems (such as integer factorization), which involve finding objects (such as prime factors) that satisfy polynomial-time-testable conditions, but for which no systematic polynomial time (P) algorithms have ever been discovered.

Random sequence generators have been constructed with the property that recognizing patterns in the sequences they produce is in principle equivalent to solving certain difficult number theoretical problems [2] (which are in the class NP, but are not NP-complete). An example is the sequence of least significant bits obtained by iterating the transformation $x' = x^2 \bmod (pq)$, where p and q are large primes (congruent to 3 modulo 4) [6]. Making predictions from this sequence is in principle equivalent to factoring the integer pq [6, 7].

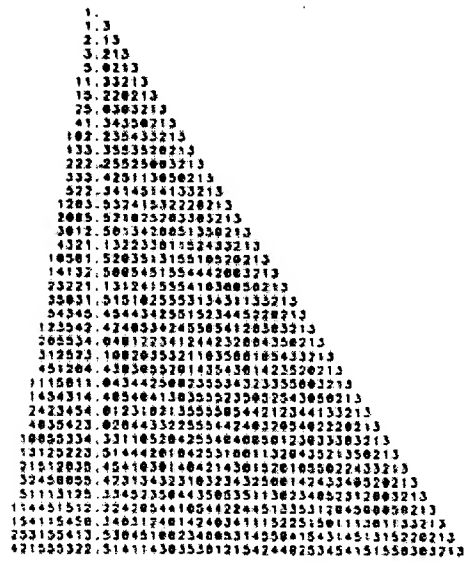
There are in fact many standard mathematical processes which are simple to perform, yet produce sequences so complicated that they seem random. An example is taking square roots of integers. Despite the simplicity of its computation, no practical statistical procedures have revealed any regularity in say the digit sequence of $\sqrt{2}$ (e.g., [8]). (Not even its normality or equidistribution has however actually been proved.) An even simpler example is multiplication by $\frac{3}{2}$, say in base 6. (2) Starting with 1, one obtains the pattern shown in Fig. 1.1 The center vertical column of values, corresponding to the leading digit in the fractional part of $(\frac{3}{2})^n$, seems random [10]. (Though again not even its normality has actually been proved.) Given the complete number obtained at a particular stage, multiplication by $(\frac{2}{3})^n$ suffices to reproduce the original seed. But given only the center column, it seems difficult to deduce the seed.

Many physical processes also yield seemingly random behaviour. In some cases, the randomness can be attributed to the effects of

external random input. Thus, for example, ``analog'' random sequence generators such as noise diodes work by sampling thermal fluctuations associated with a heat bath containing many components. Coin tossings and Roulette wheels produce outcomes that depend sensitively on initial velocities determined by complex systems with many components. It seems however that in all such cases, sequences extracted sufficiently quickly can depend on only a few components of the environment, and must eventually show definite correlations. One suspects in fact that randomness in many physical systems (probably including turbulent fluids) arises not from external random input, but rather through intrinsic mathematical processes [11]. This paper discusses the generation of random sequences by simple procedures which seem to capture many features of this phenomenon. The investigations described may not only suggest practical methods for random sequence generation, but also provide further understanding of the nature and origins of randomness in physical processes.

Figure 1.1

Successive powers of 3/2 in base 6. The leading digits in the fractional parts of these numbers form a sequence that seems random. The process of multiplication by 3/2 in base 6 corresponds to a $k=6$, $r=1$ cellular automaton rule.



Enlarge

2. Cellular Automata

A 1-dimensional cellular automaton [12, 13] consists of a line of sites with values a_i between 0 and $k-1$. These values are updated in parallel (synchronously) in discrete time steps according to a fixed rule of the form

$$a_i' = \phi(a_{i-r}, a_{i-r+1}, \cdots, a_{i+r}). \tag{2.1}$$

Much of this paper is concerned with the study of a particular $k=2$, $r=1$ cellular automaton, described in Section 3.

For mathematical purposes, it is often convenient to consider cellular automata with an infinite number of sites. But practical implementations must contain a finite number of sites N. These are typically arranged in a circular register, so as to have periodic boundary conditions, given in the $r=1$ case by

$$\begin{aligned} a_1' &= \phi(a_N, a_1, a_2) \\ a_N' &= \phi(a_{N-1}, a_N, a_1). \end{aligned} \tag{2.2}$$

It is also possible to arrange the sites in a feedback shift register (cf. [4]), with boundary conditions

$$\begin{aligned} a_1' &= \phi(\phi(a_2, a_3, a_4), \phi(a_3, a_4, a_5), a_1), \\ a_2' &= \phi(\phi(a_3, a_4, a_5), a_1, a_2). \end{aligned} \tag{2.3}$$

Cellular automata can be considered as discrete approximations to partial differential equations, and used as direct models for a wide variety of natural systems (e.g., [14]). They can also be considered as discrete dynamical systems corresponding to continuous mappings on the Cantor set (e.g., [15]). Finally they can be viewed as computational systems, whose evolution processes information contained in their initial configurations (e.g., [16]).

Despite the simplicity of their construction, cellular automata are found to be capable of diverse and complex behaviour. Figure 2.1 shows some patterns generated by evolution according to various cellular automaton rules, starting from typical disordered initial

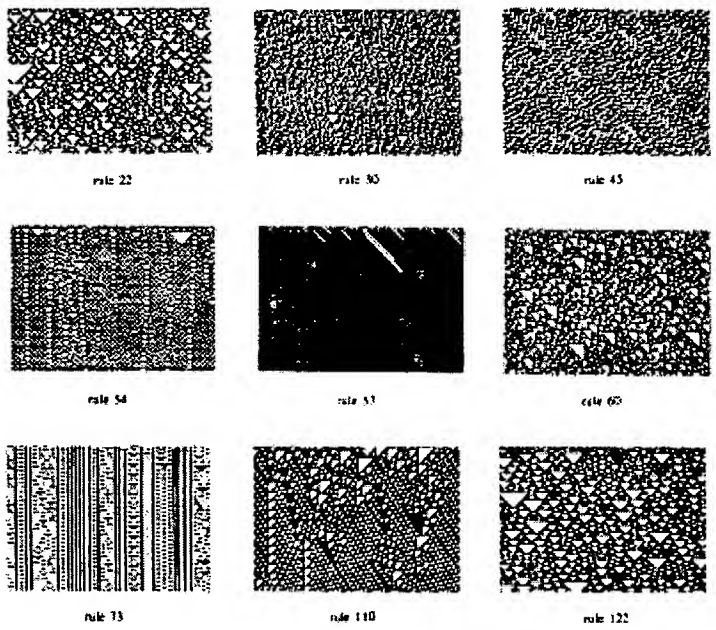
conditions. Four basic outcomes are seen [15]: (1) the pattern becomes homogeneous (fixed point), (2) the pattern degenerates into simple periodic structures (limit cycles), (3) the pattern is aperiodic, and appears chaotic, and (4) complicated localized structures are produced. The first two classes of cellular automata yield readily predictable behaviour, and show no seemingly random elements. But the third class gives rise to behaviour that is more complex. They can produce patterns whose features cannot readily be predicted in detail, and in fact often seem completely random. Such cellular automata can be used as models of randomness in nature. They can also be considered as abstract mathematical systems, and used for practical random sequence generation.

Figure 2.1 shows patterns produced by evolution according to various cellular automaton rules, starting from typical disordered initial conditions, in which the value of each site is randomly chosen to be zero or one. Figure 2.2 shows some patterns obtained instead by evolution from a very simple initial condition containing a single nonzero site. With such simple initial conditions, some class 3 cellular automata yield rather simple patterns, which are typically periodic or at least self similar (almost periodic). There are nevertheless class 3 cellular automata which yield complex patterns, even from simple initial states. Their evolution can intrinsically produce apparent randomness, without external input of random initial conditions. It is such ``autoplectic" systems [11] which seem most promising for explaining randomness in nature, or for use as practical random sequence generation procedures.

Many class 3 cellular automata seem to perform very complicated transformations on their initial conditions. Their evolution thus corresponds to a complicated computation. But any predictions of the cellular automaton behaviour must also be obtained through computations. Effective predictions require computations that are more sophisticated than those corresponding to the cellular automaton evolution itself. One suspects however that the evolution of many class 3 cellular automata is in fact computationally as sophisticated as that of any (physically realizable) system can be [18, 19]. It is thus ``computationally irreducible," and its outcome can effectively be found only by direct simulation or observation. There are no general computational shortcuts or finite mathematical formulae for it. As a consequence, many questions concerning infinite time or infinite size limits cannot be answered by bounded computations, and must be considered formally undecidable. In addition, questions about finite time or finite size behaviour, while ultimately computable, may be computationally intractable, and could require, for example, exponential time computations.

Figure 2.1

Patterns generated by evolution of various $k = 2$, $r = 1$ cellular automata from disordered initial states. Successive lines give configurations obtained on successive time steps, with white and black squares representing sites with values 0 and 1 respectively. The coefficient of 2^i in the binary decomposition of each rule number gives the value of the function ϕ in Eq. (2.1) for the neighbourhood whose site values form the integer i (cf. [17]).



Enlarge

Most class 3 cellular automata are expected to be computationally irreducible. A few rules however have special simplifying features which make predictions and analysis possible. One class of such rules are those for which the function ϕ is linear (modulo k) in the a_{i+j}. Such cellular automata are analogous to linear feedback shift registers [4]. An example with $k = 2$ is

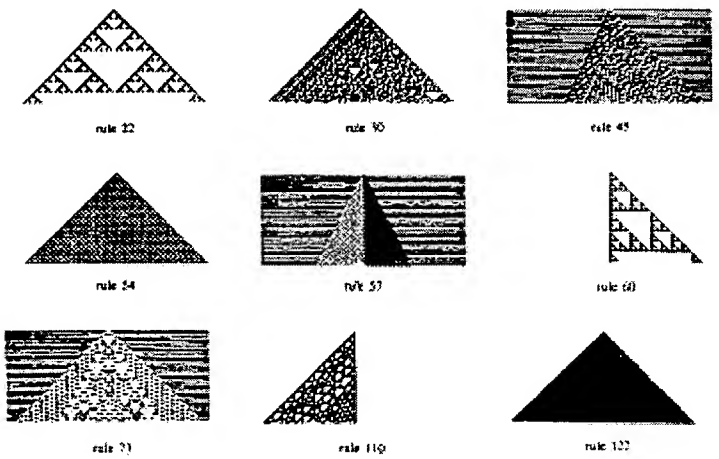
$$(2.4) \qquad a_i' = (a_{i-1} + a_i) \bmod 2 = (a_{i-1} \text{ XOR } a_i),$$

where XOR stands for exclusive disjunction (this is rule number 60 in the scheme of [17]). Linear cellular automata satisfy a superposition principle, which implies that patterns generated with arbitrary initial states can be obtained as appropriate superpositions of the self-similar pattern produced with a single nonzero initial site (as illustrated in Fig. 2.2). As a result, it is possible to give a complete algebraic description of the behaviour of the system [20], and to deduce the outcome of its evolution by a much reduced computation.

Most class 3 cellular automata are however nonlinear. No general methods to predict their behaviour have been found, and from their likely computational irreducibility one expects that no such methods even in principle exist. In studying such systems one must therefore to a large extent forsake conventional mathematical techniques and instead rely on empirical and experimental mathematical results.

Figure 2.2

Patterns generated by evolution of various $k = 2$, $r = 1$ cellular automata from an initial state containing a single nonzero site. Complex patterns are seen to be produced even with such simple initial conditions.



Enlarge

Previous section | Next section

3. A Random Sequence Generator

There are a total of $2^{2^3} = 256$ cellular automaton rules that depend on three sites, each with two possible values ($k = 2$, $r = 1$). Among these are several linear rules similar to that of Eq. (2.4). But the two rules that seem best as random sequence generators are nonlinear, and are given by

(3.1a) $a_i' = a_{i-1}$ XOR $(a_i$ OR $a_{i+1})$

or, equivalently,

(3.1b) $a_i' = (a_{i-1} + a_i + a_{i+1} + a_i a_{i+1}) \bmod 2$

(rule number 30 [17]; equivalent to rule 86 under reflection), and

(3.2a) $a_i' = a_{i-1}$ XOR $(a_i$ OR $(\text{NOT } a_{i+1}))$

or

(3.2b) $a_i' = (1 + a_{i-1} + a_{i+1} + a_i a_{i+1}) \bmod 2$

(rule 45; reflection equivalent to rule 75). Here XOR stands for exclusive disjunction (addition modulo two); OR for inclusive disjunction (Boolean addition), and NOT for negation. The patterns obtained by evolution from a single nonzero site with each of these rules were shown in Fig. 2.2. It is indeed remarkable that such complexity can arise in systems of such simple construction. A first indication of their potential for random sequence generation is the apparent randomness of the center vertical column of values in the patterns of Fig. 2.2.

This paper concentrates on the cellular automaton of Eq. (3.1). The methods used carry over directly to the cellular automaton of Eq. (3.2), but some of the results obtained in this case are slightly less favourable for random sequence generation.

The cellular automaton rule (3.1) is essentially nonlinear. Nevertheless, its dependence on a_{i-1} is in fact linear. This feature (termed ``left permutivity'' in [21], and also studied in [22]) is the basis for many of its properties. In the form (3.1), the rule gives the new value a_i' of a site in terms of the old values a_{i-1}, a_i and a_{i+1}. But the linear dependence on a_{i-1} allows the rule to be rewritten as

(3.3) $$a_{i-1} = a_i' \text{ XOR } (a_i \text{ OR } a_{i+1}),$$

giving a_{i-1} in terms of a_i', a_i and a_{i+1}. This relation implies that the spacetime patterns shown, for example, in Figs. 2.1 and 2.2 can be found not only by direct time evolution according to (3.1) from a given initial configuration, but also by extending spatially according to (3.3), starting with the temporal sequence of values of two adjacent sites.

Random sequences are obtained from (3.1) by sampling the values that a particular site attains as a function of time. In practical implementations, a finite number of sites are considered, and are typically arranged in a circular register. Given almost any initial ``seed'' configuration for the sites in the register, a long and seemingly random sequence can apparently be obtained. This paper discusses several approaches to the analysis of the cellular automaton (3.1) and the sequences it produces. While little can rigourously be proved, the overwhelming weight of evidence is that the sequences indeed have a high degree of randomness.

Previous section | Next section

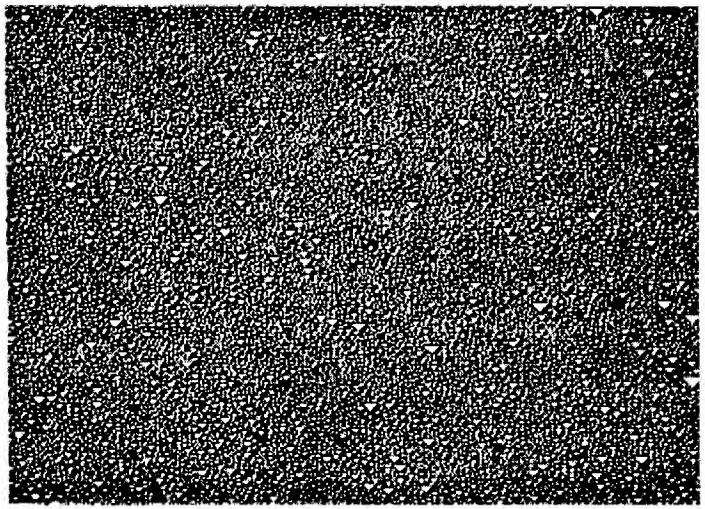
4. Global Properties

This section considers the behaviour of the cellular automaton (3.1) starting from all possible initial states. The basic approach is to count the possible sequences and patterns that can occur, and to characterize them using methods from dynamical systems theory (e.g., [23]). The next section discusses the behaviour obtained by evolution from particular initial configurations. For purposes of simplicity, this section concentrates on the infinite size limit; Section 9 considers finite size effects.

Figure 4.1 shows a spacetime pattern produced by evolution according to (3.1) starting from a typical disordered initial state. While definite structure is evident, one may suspect that a single line of sites at any angle in the pattern can have an arbitrary sequence of values. Below we shall show that this is in fact the case: given an appropriate initial condition, any sequence can be generated in an infinite cellular automaton with the rule (3.1).

Figure 4.1

Pattern produced by evolution according to the cellular automaton rule (3.1) from a typical disordered initial state.



Enlarge

The rule (3.1) can be considered as a mapping from one (say infinite) cellular automaton configuration to another. An important property of this mapping is that it is surjective or onto. Any configuration A can thus always be obtained as the image of some configuration A^-, according to $A = \phi A^-$. A possible configuration A^- (not necessarily unique) can be found by starting with a candidate pair of site values, then extending to the left using Eq. (3.3). So if all possible initial configurations are considered, then any configuration can be generated at any time step. Thus with appropriate initial conditions, any spatial sequence of site values can be produced.

Every length k spatial sequence of site values that occurs is determined by a length $k+2$ sequence on the previous time step. The surjectivity of the rule (3.1) implies that such a predecessor exists for any length k sequence. But Eq. (3.3) also implies that there are exactly four predecessors for any sequence. Given values a_i, a_{i-1}, and so on, in one sequence, the values a_{i+1}^- and a_i^- in its predecessor can be chosen in all the four possible ways; in each case the remaining a_{i-j}^- are then uniquely determined by Eq. (3.3). Thus starting from an ensemble that contains all possible (infinite) cellular automaton configurations with equal probabilities, each configuration will be generated with equal probability throughout the evolution of the cellular automaton, and so every possible spatial sequence of a particular length will occur with equal frequency.

One may also consider sequences of values attained by a single site as a function of time. Starting from an initial ensemble which contains all configurations with equal probabilities, all such sequences again occur with equal frequencies. For, given any temporal sequence, iteration of Eq. (3.3) yields an equal number of initial configurations which evolve to it. The same is true for sequences of site values on lines at any angle in the spacetime pattern.

Entropies provide characterizations of the number of possible sequences that occur. First, let the number of distinct length n blocks in these sequences be $N(n)$, and let the ith such sequence appear with probability p_i. Then the topological entropy of the sequence is given by (e.g., [15])

$$(4.1) \qquad s = \lim_{n\to\infty} \frac{1}{n} \log_2 N(n),$$

and the measure entropy by

$$(4.2) \qquad s_\mu = \lim_{n\to\infty} \frac{-1}{n} \sum_i^{2^n} p_i \log_2 p_i.$$

If the cellular automaton configurations are considered as elements of a Cantor set, then these entropies give respectively the Hausdorff (strictly Kolmogorov) and measure dimensions of this set. If the sequences are considered as ``messages," then the entropies give respectively their capacity and Shannon information content.

For the cellular automaton of Eq. (3.1), all possible sequences occur with equal probabilities (given an equal probability initial ensemble) so both entropies are maximal:

$$(4.3) \qquad s_\mu = s = 1.$$

Any reduction in entropy would reveal redundancy in the sequences, and would imply a lack of randomness. Equation (4.3) is thus a necessary (though not sufficient) condition for randomness. (It is related to statistical test A of Section 10 and Appendix A.)

Although Eq. (4.3) implies that all possible sequences of values for single sites can occur along any spacetime direction, the deterministic nature of the cellular automaton rule (3.1) implies that only certain spacetime patches of values can occur. In fact, all the site values in a particular patch are completely determined by the values that appear on its upper, left and right boundaries. Once these boundaries are specified, the values of remaining sites in the patch are redundant, and can be found simply by applying (3.1) and (3.3).

In general the degree of redundancy in such spacetime patterns can be characterized by the invariant topological and measure entropies for the cellular automaton mapping, given by (e.g., [15, 24])

$$(4.4) \qquad \mathrm{h} = \lim_{X\to\infty} \lim_{T\to\infty} \frac{1}{T} \log_2 N(X, T)$$

and

(4.5) $$h_\mu = \lim_{X\to\infty} \lim_{T\to\infty} \frac{-1}{T} \sum_{i=1}^{2^{XT}} p_i \log_2 p_i,$$

where $N(X,T)$ gives the total number of distinct $X \times T$ spacetime patches of site values that occur, and the p_i give their probabilities.

It is clear from the locality of the rule (3.1) that

(4.6) $$h_\mu \leq h \leq 2.$$

A calculation based on the method of [25] in fact shows that (3)

(4.7) $$h_\mu \lesssim 1.20.$$

Hence a knowledge of the time sequences of values of about 1.2 sites suffice in principle to determine the values of all other sites. In practice however the function which gives the initial configuration in terms of these temporal sequences seems rapidly to become intractably complicated, as discussed in Section 7.

Previous section | Next section

5. Stability Properties

Section 4 considered properties of possible patterns generated by evolution with the cellular automaton rule of Eq. (3.1), starting from all possible initial configurations. This section considers the change in the patterns produced by small perturbations in the initial state. Figure 5.1 shows the differences resulting from reversal of a single site value in a typical disordered initial configuration. The region affected increases in size with time, reflecting the instability of the patterns generated.

This instability implies that information on localized changes eventually propagates throughout the cellular automaton. The rates of information transmission to the left and right are determined by the slopes of the difference pattern in Fig. 5.1. These in turn give left and right Lyapunov exponents λ_L and λ_R for the cellular automaton evolution [15, 26]. (The sequence of site values in a configuration, starting from a particular point, can be represented as a real number. Linear growth of the difference pattern in Fig. 5.1 then implies exponential divergence of the numbers representing nearby configurations.)

The form of the cellular automaton rule (3.1) immediately implies that

(5.1) $$\lambda_R = 1.$$

For consider a configuration in which the difference pattern has reached site -1. Whatever the current values of sites 0 and 1, the XOR in (3.1) leads to a change in the new value of site 0. The value (5.1) is the maximum allowed by the locality of the rule (3.1).

Empirical measurements suggest that the left-hand side of the difference pattern expands at an asymptotically linear rate, with a slope [45]

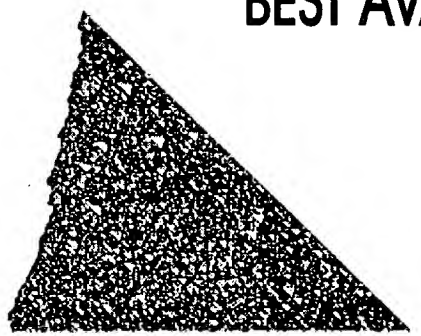
(5.2) $$\lambda_L = (0.2428 \pm 0.0003).$$

A simple statistical estimate for λ_L can be given. Consider a pair of configurations for which the front of the difference pattern has reached site 0. As a first approximation, one may assume that the motion of this front depends only on the neighbouring values a_{-1} and a_{+1}, where, by construction, a_{-1} is the same for the two configurations. When $a_{-1} = 0$, the front advances (left) by one site,

independent of the values of the a_i. When $a_{-1} = 1$, the front remains stationary if the a_{+1} for the two configurations are equal, and retreats by one site if they are unequal. If possible sets of site values occurred with equal probabilities, the front should thus follow a biased random walk, advancing at average speed $1/4$. In practice, however, Fig. 5.1 shows that the front can retreat by many sites in a single time step. This occurs when the cellular automaton rule yields the same image for multiple site value sequences, as for say 10100 and 11001. Such phenomena make the probabilities for different difference patterns unequal, and invalidate this purely statistical approach discussed. (The values of λ_L obtained in this approach by considering the effects of between 1 and 5 sites on the right are 0.25, 0.1875, 0.15625, 0.140625 and 0.134766.)

Figure 5.1

Differences in patterns produced by evolution according to the cellular automaton rule of Eq. (3.1) from two typical disordered states which differ by reversal of the centre site value. The growth of the region of differences reflects the instability of the cellular automaton evolution.



Enlarge

The result (5.2) gives the average speed of the left-hand side of the difference pattern. As the random walk interpretation suggests, however, one can choose initial configurations for which a single site change leads to differences which expand at speed 1 on the left. In general, one can construct the analog of a Green's function, giving the probability that a site at a particular position and time will be affected by an initial perturbation. This function is nonzero within a ``light cone'' with edges expanding at speed 1. It appears to be uniform on the right-hand side. But on the left-hand side, it appears to be determined by a diffusion equation which gives the average behaviour of the biased random walk. The difference pattern can thus extend beyond the line given by Eq. (5.2), but with an exponentially damped probability.

Lyapunov exponents measure the rate of information transmission in cellular automata, and provide upper bounds on entropies, which measure the information content of patterns generated by cellular automaton evolution. For surjective cellular automata it can

be shown, for example, that [15]

(5.3) $h_{KS} \leq (\lambda_L + \lambda_R)$,

consistent with Eq. (4.6) and (5.2). The existence of positive Lyapunov exponents is a characteristic feature of class 3 cellular automata.

The difference pattern of Fig. 5.1, and the related Green's function, measure the effect of initial perturbations on the values of individual sites. In studying random sequence generation, one must also consider the effect of such perturbations on time sequences of site values, say of length t. These sequences are always completely determined from the initial values of $2t+1$ sites. But not all these initial values necessarily affect the time sequences. A change in any of the $t+1$ left-hand initial sites necessarily leads to a change in at least one element of the time sequence. But some changes in the t right-hand initial sites have no effect on any element of the time sequence. It seems that the probability for a particular initial site to affect the time sequence decreases exponentially with distance to the right. The average number of sites on the right which affect the time sequence is found to be approximately $0.26 + 0.19t$. Thus the total number of initial sites on which a length t time sequence depends is on average approximately $1.91 + 1.19t$. This result is presumably related to the entropy (4.6).

Previous section | Next section

6. Particular Initial States

Sections 4 and 5 have discussed some properties of the patterns produced by evolution according to Eq. (3.1) from generic initial conditions. This section considers evolution from particular special initial configurations.

Figure 6.1 shows on two scales the pattern produced by evolution from a configuration containing a single nonzero site. (This could be considered a difference pattern for the special time-invariant state in which all sites have value zero.) Remarkable complexity is evident.

There are however some definite regularities. For example, diagonal sequences of sites on the left-hand side of the pattern are periodic, with small periods. In general, the value of a site at a depth N from the edge of the pattern depends only on sites at depths N or less; all the other sites on which it could depend always have value 0 because of the initial conditions given. As a consequence, the sites down to depth N are independent of those deeper in the pattern, and in fact follow a shifted version of the cellular automaton rule (3.1), with boundary conditions that constrain two sites at one end to have value zero. Since such a finite cellular automaton has a total of 2^N possible states, any time sequence of values in it must have a period of at most 2^N. The corresponding diagonal sequences in the pattern of Fig. 6.1 must therefore also have periods not greater than 2^N.

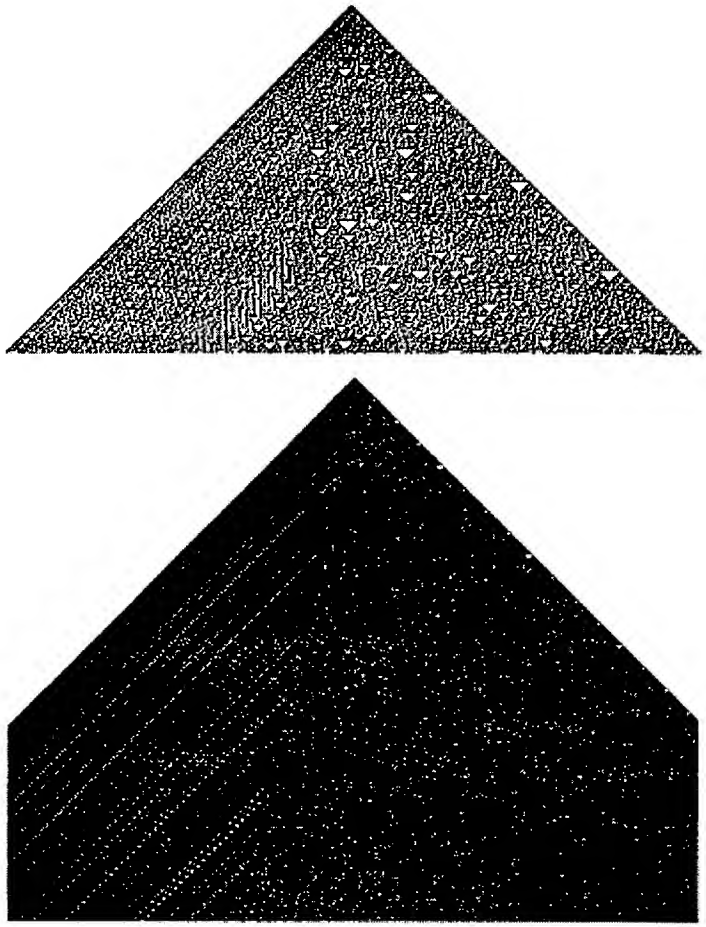
Table 6.1 gives the actual periods of diagonal sequences found at various depths on the left- and right-hand sides of the pattern in Fig. 6.1. These are compared with those for the self-similar pattern shown in Fig. 2.2 generated by evolution according to the linear cellular automaton rule (2.4).

The short periods on the left-hand side of the pattern in Fig. 6.1 are related to the high degree of irreversibility in the effective cellular automaton rule for diagonal sequences in this case [27]. Starting with any possible initial configuration, this cellular automaton always yields cycles with period 2^j. The maximum value of j increases very slowly with N, yielding maximum cycle lengths which increase in jumps, on average slower than linearly with N. (Between the N values at which the maximum cycle length increases, a single additional cycle of maximal length seems to be added each time N increases by one. The total number of cycle states thus increases at most quadratically with N, implying an increasing degree of irreversibility.) The actual sequences that occur near the left-hand boundary of the pattern in Fig. 6.1 correspond to a particular set of those possible in this effective cellular

automaton. In a first approximation, they can be considered uniformly distributed among possible N-site configurations, and their periods increase very slowly with N.

Figure 6.1

Patterns generated by evolution for 250 and 2000 generations, respectively, according to the cellular automaton rule (3.1) from an initial state containing a single nonzero site. (The second pattern was obtained by Jim Salem using a prototype Connection Machine computer.)



Enlarge

Table 6.1

Period lengths for diagonal sequences in patterns generated by evolution from a single nonzero site according to the cellular automaton rules of Eqs. (3.1) and (2.4). π_R and π_L signify respectively periods for diagonal sequences on the right and left of the patterns, at the specified depth. (The entries left blank were not found.)

	CA 30		CA 60
Depth	π_R	π_L	π_R
0	1	1	1
1	2	1	2
2	2	1	4
3	4	2	4
4	8	1	8
5	8	2	8
6	16	2	8
7	32	1	8
8	32	4	16
9	64	1	16
10	64	4	16
11	64	4	16
12	64	4	16
13	64	4	16
14	64	4	16
15	128	4	16
16	256	4	32
32		8	64
64		4	128
128		8	256
256		8	512
512		16	1024
1024		16	2048

The effective rule for the right-hand side diagonal pattern in Fig. 6.1 is a shifted version of Eq. (3.1)

(6.1a) $$a_i' = a_i \text{ XOR } (a_{i+1} \text{ OR } a_{i+2}),$$

with boundary conditions

(6.1b) $$a_{N-1}' = a_{N-1} \text{ XOR } a_N,$$
$$a_N' = a_N.$$

This system is exactly reversible: all of its 2^N possible configurations have unique predecessors. All the configurations thus lie on

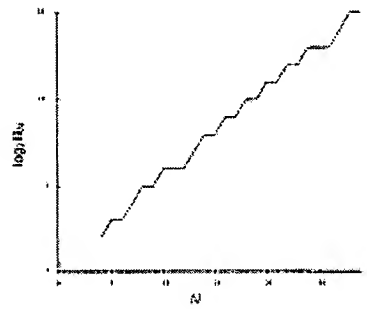
cycles, and again the cycles have periods of the form 2^i. Figure 6.2 shows the lengths of longest cycles as a function of N. These lengths increase roughly exponentially with N; a least squares fit to the data of Fig. 6.2 yields

(6.2) $$\log_2 \Pi_N \simeq 0.5(N+1).$$

This length is small compared to the total number of states 2^N; few states in fact lie on such longest cycles. Nevertheless, the periods of the right-hand diagonal sequences in Fig. 6.1 do seem to increase roughly exponentially with depth, as suggested by Table 6.1.

Figure 6.2

Maximal period lengths Π_N for the effective cellular automaton which gives the right-hand diagonal sequences in Fig. 6.1 down to depth N. Points plotted at integer N are joined for pictorial purposes.



Enlarge

The boundary in Fig. 6.1 between regular behaviour on the left and irregular behaviour on the right seems to be asymptotically linear, and to move to the left with speed 0.25. A statistical argument for this result can be given in analogy with that for Eq. (5.2). Each site at depth d on the left-hand side of the pattern could in principle be affected by sites down to depth d arbitrarily far up in the pattern. In practice, however, it is unaffected by changes in sites outside a cone whose boundary propagates at speed $\lambda_L \simeq 0.25$. Thus the irregularity on the right spreads to the left only at this speed.

While diagonal sequences at angles ± 1 in Fig. 6.1 must ultimately become periodic, sequences closer to the vertical need not. In fact, no periodicity has been found in any such sequences. The center vertical (i.e., temporal) sequence has, for example, been tested up to length $2^{19} \simeq 5 \times 10^5$, and no periodicity is seen. One can prove in fact that only one such vertical sequence (obtained from any initial state containing a finite number of nonzero sites) can possibly be periodic [22]. For if two sequences were both periodic, then

it would follow that all sequences to their right must also be, which would lead to a contradiction at the edge of the pattern.

Not only has no periodicity been detected in the center vertical sequence of Fig. 6.1; the sequence has also passed all other statistical tests of randomness applied to it, as discussed in Section 10.

While individual sequences seem random, there are local regularities in the overall pattern of Fig. 6.1. Examples are the triangular regions of zero sites. Such regularities are associated with invariants of the cellular automaton rule.

The particular configuration in which all sites have value 0 is invariant under the cellular automaton rule of Eq. (3.1). As a consequence, any string of zeroes that appears can be corrupted only by effects that propagate in from its ends. Thus each string of zeroes that is produced leads to a uniform triangular region.

Table 6.2 and Fig. 6.3 give other configurations which are periodic under the rule (3.1). (They can be considered as invariant under iterations of the rule.) Again, any string that contains just the sequences in these configurations can be corrupted only through end effects, and leads to a regular region in spacetime patterns generated by Eq. (3.1).

Table 6.2

Configurations periodic under the cellular automaton mapping (3.1) consist of infinite repetitions of the elements given. Notice that the four elements given for period four correspond simply to different phases in a cycle. The patterns generated by these periodic configurations are shown in Fig. 6.3.

Period	Element
1	0 01
3	000011111001
4	0000001 0000111 0010011 0111111

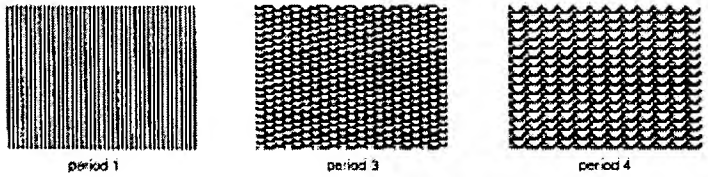
In general, there is a finite set of configurations with any particular period p under a permutive cellular automaton rule such as (3.1). The configurations may be found by starting with a candidate length $2p$ string, then testing whether this and the string it yields through Eq. (3.3) on the left are in fact invariant under ϕ^p. The string to be tested need never be longer than 2^{2p}, since such a string can contain all possible length $2p$ strings. Thus the periodic configurations consist of repetitions of blocks containing 2^{2p} or less site values. (For an arbitrary cellular automaton rule, the set of invariant configurations forms a finite complement language which contains in general an infinite number of sequences with the constraint that certain blocks are excluded [16].)

The pattern in Fig. 6.1 can be considered the effect of a single site ``defect'' in the periodic pattern resulting from a configuration with all sites 0. Figure 6.4 shows difference patterns produced by single site defects in the other periodic configurations of Table 6.2 and Fig. 6.3.

The periodic configurations of Table 6.2 and Fig. 6.3 can be viewed as special states in which the cellular automaton of Eq. (3.1) behaves just like the identity rule. Concatenations of other blocks could simulate other cellular automata: one block might correspond to a value 0 site, and another to a value 1 site in the effective cellular automaton. Some cellular automata (such as that of Eq. (2.4)) simulate themselves under such ``blocking transformations,'' and thus evolve to self-similar patterns. The cellular automata of Eqs. (3.1) and (3.2) are unique among $k = 2,\ r = 1$ rules in simulating no other rules, at least with blocks of length up to eight [14].

Figure 6.3

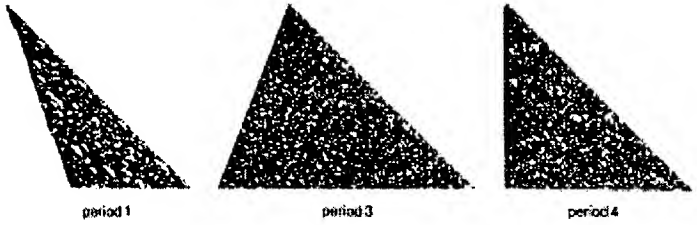
Periodic patterns for the cellular automaton rule of Eq. (3.1). The form of these patterns is given in Table 6.2.



Enlarge

Figure 6.4

Patterns produced by evolution according to the cellular automaton rule (3.1) by single site initial defects in the periodic patterns of Fig. 6.2 and Table 6.2.



Previous section | Next section

7. Functional Properties

Cellular automaton rules such as (3.1) can be considered as functions ϕ which map three Boolean values to one. Iterations of these rules for say t steps correspond to functions of $2t+1$ Boolean values. The complexity of these functions reflects the intrinsic complexity of the cellular automaton evolution.

The complexity of a Boolean function can be characterized by the number of logic gates that would be needed to evaluate it with a particular kind of circuit, or the number of terms that it would have in a particular symbolic representation. Explicit evolution according to the cellular automaton rule (3.1) corresponds to a circuit with $O(t^2)$ components and depth t. But for purposes of comparison, it is convenient to consider fixed depth representations. One such representation is disjunctive normal form (DNF), in which the function is written as a disjunction of conjunctions. A two-level circuit can be constructed in direct correspondence with this form (as programmable logic arrays often are).

For the function of Eq. (3.1), the DNF is

$$\phi(a_{-1}, a_0, a_1) = (\overline{a_{-1}} a_0) + (a_{-1} \overline{a_0}\ \overline{a_1}) + (\overline{a_{-1}} a_1), \tag{7.1}$$

where $+$ stands for OR, concatenation for AND, and bar for NOT. Notice that by using in addition an XOR operation, Eq. (3.1) itself gives a shorter form for this function.

The general problem of finding the absolute shortest representation for an arbitrary Boolean function, even in DNF, is NP-complete (e.g., [5]), and so presumably requires an exponential time computation. But a definite approximation can be found in terms of ``prime implicants'' (e.g., [28]). A Boolean function of n variables can be considered as a colouring of the Boolean n-cube. Prime implicants give the hyperplanes (with different dimensions) in the n-cube which must be superimposed to obtain the region with value 1. Each prime implicant can thus be used as a term in a DNF for the function. The number of prime implicants required gives a measure of the total number of ``holes'' in the colouring of the n-cube, and thus of the complexity of the function.

Table 7.1

Number of terms in disjunctive normal form Boolean expressions corresponding to iterations of the mappings (3.1) (CA30) and (2.4) (CA60). P.I. gives the number of prime implicants; min. the number of terms obtained by [29]. (The two numbers are the equal in the case of Eq. (2.4).)

	CA30		CA60
t	P.I.	Min.	P.I./Min.
1	3	3	2
2	9	7	2
3	23	17	8
4	76	41	2
5	185	105	8
6	666	272	8

The minimal DNF obtained with prime implicants for the function corresponding to two iterations of the cellular automaton mapping (3.1) is

$$\begin{aligned}\phi^2(a_{-2}, a_{-1}, a_0, a_1, a_2) = {} & (\overline{a_{-2}}\,\overline{a_{-1}}\,\overline{a_0}\,a_1\overline{a_2}) + (\overline{a_{-2}}a_{-1}a_0a_1\,\overline{a_2}) \\ & + (a_{-2}\overline{a_{-1}}a_0a_1\,\overline{a_2}) + (a_{-2}a_{-1}a_0\overline{a_1}\,\overline{a_2}) \\ & + (a_{-2}\overline{a_{-1}}\,\overline{a_1}\,\overline{a_2}) + (\overline{a_{-2}}\,\overline{a_{-1}}\,\overline{a_0}\,a_2) \\ & + (a_{-2}\overline{a_{-1}}a_0a_2) + (\overline{a_{-2}}a_{-1}a_0a_2) + (a_{-2}a_{-1}\,\overline{a_0}).\end{aligned} \tag{7.2}$$

Table 7.1 gives the number of prime implicants for successive iterations of the mapping (3.1). These results are plotted in Fig. 7.1. For arbitrary Boolean functions of $2t+1$ variables, the number of prime implicants could increase like 4^t. In practice, however, a least squares fit to the data of Table 7.1 suggests growth like $4^{0.77t}$.

Various efficient methods are known to find DNF that are somewhat simpler than those obtained using prime implicants. With one such method [28, 29], the DNF of Eq. (7.2) can be reduced to

$$\begin{aligned}\phi^2(a_{-2}, a_{-1}, a_0, a_1, a_2) = {} & (\overline{a_{-2}}\,\overline{a_{-1}}\,\overline{a_0}\,a_1) + (\overline{a_{-2}}\,a_{-1}a_0a_1) \\ & + (\overline{a_{-2}}\,\overline{a_{-1}}\,\overline{a_0}\,a_2) + (\overline{a_{-2}}\,a_{-1}a_0a_2) \\ & + (a_{-2}\overline{a_1}\,\overline{a_2}) + (a_{-2}\overline{a_{-1}}\,a_0) + (a_{-2}a_{-1}\overline{a_0}).\end{aligned} \tag{7.3}$$

The sizes of the minimal DNF obtained by this method for iterations of Eq. (3.1) are shown in Table 7.1 and Fig. 7.1. They are seen to grow more slowly than those obtained with prime implicants; the data given are however again fit by exponential growth like $4^{0.65t}$.

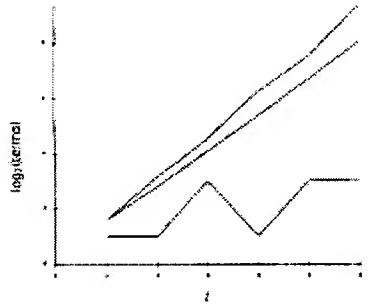
Table 7.1 and Fig. 7.1 also give the size of the minimal DNF for iterations of the linear cellular automaton mapping (2.4). This number remains much smaller, apparently increasing like $2^{2\#_1(t)-1} < t^2$, where $\#_1(t)$ gives the number of ones in the binary

representation for the integer t (cf. [30]).

The rapid increase in the size of the minimal DNF found for iterations of Eq. (3.1) indicates the increasing computational complexity of determining the result of evolution according to (3.1), and supports the conjecture of its computational irreducibility. (Note however that even the parity function cannot be computed by any DNF, or in general fixed-depth, circuit of polynomial size [31].)

Figure 7.1

Number of terms in disjunctive normal form Boolean expressions for t step iterations of the mappings (3.1) and (2.4). The upper curve gives the number of prime implicants for iterations of Eq. (3.1). The next curve gives the minimal number of terms obtained in this case using [29]. The lowest curve gives the minimal number of terms for the linear cellular automaton mapping (2.4).



Enlarge

Equation (7.3) gives the function which determines the value of a single site after two iterations of the cellular automaton rule (3.1). One can also construct a function which gives the length t sequence of values of a particular site attained through time by evolution from a given length $2t+1$ initial sequence. The minimal DNF representation for this function is found (using [29]) to grow in size approximately as $2^{1.36t}$.

The results of Table 7.1 and Fig. 7.1 concern the difficulty of finding the outcome of cellular automaton evolution according to Eq. (3.1) from a given initial state. One may also consider the problem of deducing the initial state from time sequences of site values produced in the evolution. Given say t steps in the time sequence of values for two adjacent sites, the initial configuration up to t sites to the left can be deduced directly by iteration of Eq. (3.3). The combinatorial results of Section 4 indicate in fact that only about 1.2 such temporal sequences should on average be required. And in principle from a single sufficiently long temporal sequence, it should be possible to deduce a complete initial configuration for a finite cellular automaton. In practice, however, the

necessary computation seems to become increasingly intractable as the size of the system increases.

Given a particular temporal sequence, say at position 0, Eq. (3.3) uniquely determines the values of all sites in a triangle to the left as a function of values in the temporal sequence at position 1. The number of values in the position 1 temporal sequence on which a given site depends varies with the form of the position 0 sequence [32]. For example, if the position 0 sequence consists solely of ones, then the whole triangle of sites is completely determined, entirely independent of the position 1 sequence. Table 7.2 gives some results from considering the dependence of the site value a_{-t} at position $-t$ (the apex of the triangle) on the position 1 sequence, for all 2^t possible position 0 sequences. The number of values in the position 1 sequence on which a_{-t} depends seems to be roughly Poisson distributed, with a mean that grows like $0.4t$, as shown in Fig. 7.2. This is consistent with the combinatorial result (4.6).

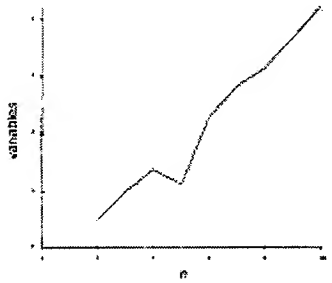
Table 7.2

Properties of Boolean expressions for leftmost initial site values deduced from length n time sequences, obtained by evolution according to Eq. (3.1). The average number of variables appearing in the Boolean expressions is given, together with the number of prime implicants in the disjunctive normal form for the expression. The maximum number of variables which can appear is always $n-1$. (Results for $n \geq 9$ were obtained by Carl Feynman using a Symbolics 3600 LISP machine. The entries left blank were not found.)

n	⟨Var.⟩	⟨P.I.⟩	Max. P.I.
2	0.5	0.75	1
3	1	1.125	2
4	1.375	1.375	3
5	1.125	1.219	3
6	2.281	2.719	12
7	2.828	3.539	17
8	3.164	4.105	26
9	3.699		
10	4.254		

Figure 7.2

Average number of additional site values necessary to ``back-track'' and determine uniquely the initial site value a_{-n} given the sequence of values a_0 for n subsequent time steps.



Enlarge

Table 7.2 also gives some properties of the prime implicant forms for a_{-t}. It is clear that the complexity of the function that determines a_{-t} from temporal sequences grows with t, probably at an increasingly rapid rate. Again this suggests that the problem of deducing the initial sequence for evolution according to Eq. (3.1), while combinatorially possible, is computational complex.

By comparison, the corresponding problem for evolution according to the linear rule (2.4) is quite straightforward. For each possible position 0 sequence, there are only two possible forms for the dependence of a_{-t} on the position 1 sequence, and each of them involves exactly 2^{t-1} prime implicants. This simplicity can be viewed as a consequence of the algebraic structure associated with this system.

Previous section | Next section

8. Computation Theoretical Properties

The discussion of the previous section can be considered as giving a characterization of the computational complexity of iterations of the cellular automaton mapping (3.1) in a particular simple model of computation. The results obtained suggest that at least in this model, there is no shortcut method for finding the outcome of the evolution: the computations required are no less than for an explicit simulation of each time step. As discussed above, one suspects in fact that the evolution is in general computationally irreducible, so that no possible computation could find its outcome more efficiently than by direct simulation.

This would be the case if the cellular automaton of Eq. (3.1) could act as an efficient universal computer (e.g., [33]), so that with an appropriate initial state, its evolution could mimic any possible computation. In particular, it could be that the problem of finding the value of a particular site after t steps (given say a simply-specified initial state, as in Fig. 6.1) must take a time polynomial in t on any computer. (Direct simulation takes $O(t^2)$ time on a serial-processing computer, and $O(t)$ time with $O(t)$ parallel processors.) For a linear cellular automaton such as that of Eq. (2.4), this problem can be solved in a time polynomial in $\log(t)$; but for the cellular automaton of Eq. (3.1) it quite probably cannot [18].

In addition to studying cellular automaton evolution from given initial configurations, one may consider the problem of deducing configurations of the cellular automaton from partial information such as temporal sequences. In particular, one may study the computational complexity of finding the seed for a cellular automaton in a finite region from the temporal sequences it generates.

There are 2^N possible seeds for a size N cellular automaton, and one can always find which ones produce a particular sequence by trying each of them in turn. Such a procedure would however rapidly become impractical. The results in Section 7 suggest a slightly more efficient method. If it were possible to find two adjacent temporal sequences, then the seed could be found easily using Eq. (3.3). Given only one temporal sequence, however, some elements of the seed are initially undetermined. Nevertheless, in a finite size system, say with periodic boundary conditions, one can derive many distinct equations for a single site value. The site value can then be deduced by solving the resulting system of simultaneous Boolean equations. The equations will however typically involve many variables. As discussed in Section 7, the number of variables seems to be Poisson-distributed with a mean around $0.4N$.

The general problem of solving a Boolean equation in n variables is NP-complete (e.g., [5]), and so presumably cannot be solved in a time polynomial in n. In addition, it seems likely that the average time to solve an arbitrary Boolean equation is correspondingly long. To relate the problem of deducing the seed discussed above to this would however require a demonstration that the Boolean equations generated were in a sense uniformly distributed over all possibilities. Out of all 2^{2^n}-variable equations, the problem here typically involves $O(2^n)$, but these seem to have no special simplifying features. At least with the method discussed above, it is thus conceivable that the problem of deducing the seed is equivalent to the general problem of solving Boolean equations, which is NP-complete.

Previous section | Next section

9. Finite Size Behaviour

Much of the discussion above has concerned the behaviour of the cellular automaton (3.1) in the idealized limit of an infinite lattice of sites. But practical implementations must use finite size registers, and certain global properties can depend on the size and boundary conditions chosen.

The total number of possible states in a size N cellular automaton is 2^N. Evolution between these states can be represented by a finite state transition diagram. Figure 9.1 gives some examples of such diagrams for the cellular automaton of Eq. (3.1) with periodic boundary conditions, as in Eq. (2.2). Table 9.1 summarizes some of their properties. The results are seen to depend not only on the magnitude of N, but also presumably on its number theoretical properties.

Each state transition diagram contains a set of cycles, fed by trees representing transients. The cycles may be considered as ``attractors" to which states in their ``basins of attraction" irreversibly evolve.

There are many regularities in the structure of the state transition diagrams obtained from Eq. (3.1). The evolution is thus not well-approximated by a random mapping between 2^N states. Table 9.1

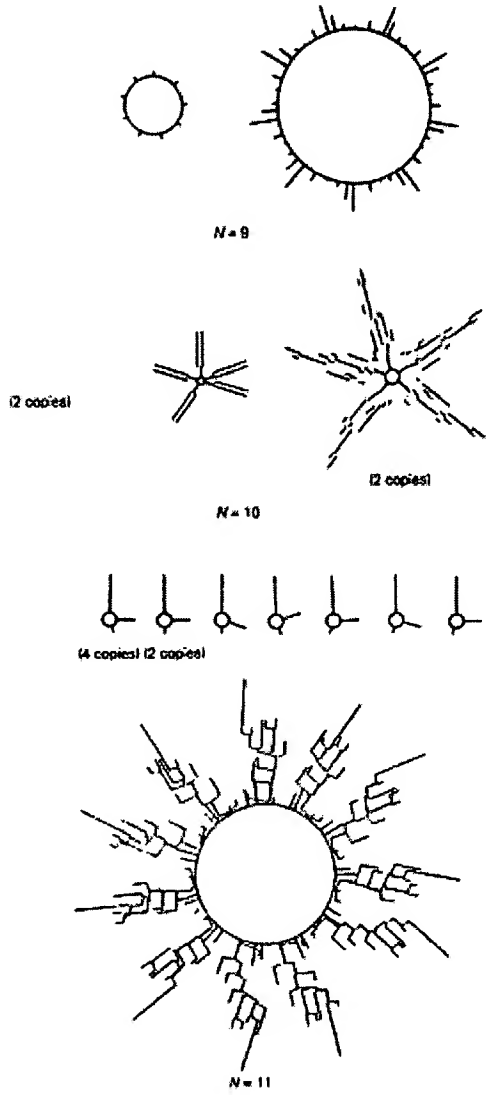
Properties of state transition diagrams for the cellular automaton rule of Eq. (3.1) in a circular register of size N. The multiplicity and length of each cycle is given, followed by the fraction of initial states which evolve to a longest cycle (size of attractor basin), the total fraction of all 2^N states which lie on cycles, and the average length of transient before a cycle is reached in evolution from an arbitrary initial state. (Results for $N \geq 16$ were obtained by Holly Peck.)

N	Cycles	Frac. longest	Cyc. frac.	⟨Transient⟩
4	$1\times 8, 3\times 1$	0.75	0.69	0.5
5	$1\times 5, 1\times 1$	0.94	0.19	4.3
6	3×1	1.00	0.05	3.3
7	$1\times 63, 7\times 4, 1\times 1$	0.60	0.72	0.4
8	$1\times 40, 1\times 8, 3\times 1$	0.88	0.20	3.1
9	$1\times 171, 1\times 72, 1\times 1$	0.81	0.48	1.1
10	$2\times 15, 1\times 5, 3\times 1$	0.82	0.04	14.8
11	$1\times 154, 11\times 17, 1\times 1$	0.76	0.17	3.3
12	$4\times 102, 1\times 8, 4\times 3, 3\times 1$	0.93	0.11	4.4
13	$1\times 832, 1\times 260, 1\times 247, 1\times 91, 1\times 1$	0.32	0.17	2.2
14	$1\times 1428, 2\times 133, 1\times 112, 2\times 84, 1\times 63, 1\times 14, 3\times 1$	0.84	0.13	2.7
15	$1\times 1455, 5\times 30, 5\times 9, 15\times 7, 4\times 5, 1\times 1$	0.93	0.05	5.7
16	$1\times 6016, 1\times 4144, 3\times 40, 1\times 8, 3\times 1$	0.50	0.16	
17	$1\times 10846, 1\times 1632, 1\times 867, 1\times 306, 1\times 136, 1\times 17, 1\times 1$	0.96	0.11	

A first observation is that most configurations have unique predecessors under the mapping (3.1) (as mentioned for infinite lattices in Section 4), so there is little branching in the state transition diagram. In fact, it can be shown [32] that a configuration has a unique predecessor unless it contains a pair of value zero sites separated by a sequence of $3n+1$ value one sites (with $n \geq 0$), or unless N is divisible by 3, and all sites have value one. In the former case, the configuration has exactly zero or two predecessors; in the latter case, it has three. The numbers of configurations with zero and two predecessors are equal when N is not divisible by 3; there are two more with zero predecessors when $3|N$. For large N, the number of configurations with zero or two predecessors behaves as [32] κ^N, where $\kappa \simeq 1.696$ is the real root of $4\kappa^3 - 2\kappa^2 - 1 = 0$. Since the total number of configurations grows like 2^N, the fraction of nodes in the state transition diagram that are branch points thus tends exponentially to zero.

Figure 9.1

State transition diagrams for configurations of cellular automata evolving according to Eq. (3.1) in circular registers of size N. Each node represents one of the 2^N possible length N configurations, and is joined by an arc to its successor under the cellular automaton mapping. Transients corresponding to trees in the graph are seen ultimately to evolve to periodic cycles. Some properties of these state transition diagrams are given in Table 9.1. (Graphics by Steve Strassmann.)



Enlarge

A second observation is that there are often many identical parts in the state transition diagrams of Table 9.1 and Fig. 9.1. This is largely a consequence of shift invariance. States in a cellular automaton with periodic boundary conditions that are related by shifts

(translations) evolve equivalently. Thus, for example, there are often several identical cycles, related by shifts in their configurations. In addition, the periods of the cycles are often divisible by N or its factors, since they contain several sequences of configurations related by shifts. The transient trees that feed each of these sequences are then identical.

The evolution of a finite cellular automaton with periodic boundary conditions is equivalent to the evolution of an infinite cellular automaton with a periodic initial configuration. Thus the results on cycle length distributions in Table 9.1 can be considered as inverse to those in Table 6.2 on configurations with given temporal periods. Cycles of lengths corresponding to these temporal periods occur whenever N is divisible by the spatial periods of these configurations. Such short cycles are absent if N has none of these factors.

For large N, the state transition diagrams for Eq. (3.1) appear to be increasingly dominated by a single cycle. This cycle is longer than the others, and its basin of attraction is large enough that most arbitrarily chosen initial states evolve to it. The low degree of branching in the transient trees implies that the points reached from arbitrary initial states should be roughly uniformly distributed around the cycle.

The shorter cycles in Table 9.1 can be considered as related to subsets of states invariant under the cellular automaton rule. With N even, for example, configurations which consist of two identical length $N/2$ subsequences can evolve only to configurations of the same type. Once such a configuration has been reached, the evolution is ``trapped" within this subset of configurations, and must yield shorter cycles. (This phenemonon also occurs for cellular automata with essentially trivial rules, such as the shift mapping $a_i' = a_i$. All states are on cycles in this case. The different cycles correspond to the possible ``necklaces" with N beads of two kinds, which are inequivalent under shifts or rotations. These necklaces in turn correspond to cyclotomic polynomials; there are $\sum_{d|N} \phi(d) 2^{N/d}$ of them, where ϕ the Euler totient function (e.g., [4]).) In general, there may exist subsets of states with certain special symmetry properties that are preserved by the cellular automaton rule. Initial states with particular, symmetrical, forms can be expected to have these properties, and thus to be trapped in subsets of state space, and to yield short cycles. For example, with $N = 36$, a configuration containing a single nonzero site evolves to a length 2844 cycle, while most initial configurations evolve to the longest cycle, with 2237472 states.

In the infinite size limit, patterns such as that of Fig. 6.1 generated by the cellular automaton of Eq. (3.1) never become periodic. But with a total of N sites, a cycle must occur after 2^N or less steps. Table 9.2 and Fig. 9.2 give the actual maximal cycle lengths Π_N found. A roughly exponential increase of Π_N with N is seen, and a least squares fit to the data of Table 9.2 yields

$$\log_2 \Pi_N \simeq 0.61(N+1). \tag{9.1}$$

Note that if the state transition diagram corresponded to an entirely random mapping between the 2^N cellular automaton states, then cycles of average length $2^{N/2}$ would be expected [34]. The cycles actually obtained are significantly longer. The exponent in Eq. (9.1) may be related to the entropy (4.6) as a result of the expansivity or instability of the mapping discussed in Section 5.

If there were very short cycles, then the sequences produced by the cellular automaton would readily be predictable. So if in fact no such prediction can be made by any polynomial time computation, the length of the cycles that occur should in general increase asymptotically faster than polynomial in N (cf. [2]). This behaviour is supported by Eq. (9.1).

If indeed the evolution of cellular automata such as (3.1) is computationally irreducible, then a complex computation may always be required to determine for example the lengths of cycles that appear. For in this case, there can effectively be no better way to find the succession of states that occur, except by explicit application of the rule (3.1). One expects in fact that the problem of finding say whether two configurations lie on the same cycle is PSPACE-complete, and so presumably cannot be solved in a time polynomial in N, but rather essentially requires a direct simulation of the cellular automaton evolution. (Note that if the lengths of the cycles studied are $O(2^M)$, where both 2^{N-M} and 2^M are large, then parallel processing is essentially of no avail in this problem.)

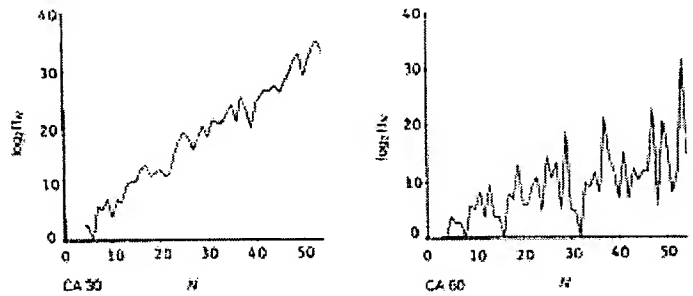
While the determination of cycle lengths and structures may be computationally intractable for cellular automata such as (3.1), it should be much easier for linear cases such as (2.4). From the algebraic theory of these systems it is possible to show for example that the maximal cycle length π_N satisfies [20]

$$\pi_N | 2^{\mathrm{ord}_N(2)} - 1, \tag{9.2}$$

where $n|m$ states that the integer n exactly divides m. Here $\mathrm{ord}_N(k)$ is the multiplicative order function, equal to the minimum integer j such that $k^j = 1 \bmod N$. This function divides the totient function $\phi(N)$ (equal to the number of integers less than N which are relatively prime to N), which is maximal for prime N. Table 9.2 and Fig. 9.2 give the actual maximal periods found in this case. Equation (9.2) rarely holds as an equality, and the π_N found are usually much shorter than the corresponding ones for the nonlinear rule (3.1).

Figure 9.2

Maximal cycle lengths π_N for the cellular automaton of Eqs. (3.1) (CA30) and (2.4) (CA60) in circular registers of size N.



Enlarge

Table 9.2

Maximum cycle lengths π_N found for the cellular automata of Eqs. (3.1) (CA30) and (2.4) (CA60) in circular registers of size N. In the former case, a selection of seeds, including single nonzero sites, were used. In the latter case, maximal length cycles are always obtained with single nonzero site seeds. The results are plotted in Fig. 9.2. (Results for $N \geq 32$ were obtained by Holly Peck and Tsutomu Shimomura with an assembly-language program on a Celerity C-1200 computer.)

	CA30		CA60	
N	Π_N	$\log_2 \Pi_N$	Π_N	$\log_2 \Pi_N$
4	8	3.0	1	0.0
5	5	2.3	15	3.9
6	1	0.0	6	2.6
7	63	6.0	7	2.8
8	40	5.3	1	0.0
9	171	7.4	63	6.0
10	15	3.9	30	4.9
11	154	7.3	341	8.4
12	102	6.7	12	3.6
13	832	9.7	819	9.7
14	1428	10.5	14	3.8
15	1455	10.5	15	3.9
16	6016	12.6	1	0.0
17	10845	13.4	255	8.0
18	2844	11.5	126	7.0
19	3705	11.9	9709	13.2
20	6150	12.6	60	5.9
21	2793	11.4	63	6.0
22	3256	11.7	682	9.4
23	38249	15.2	2047	11.0
24	185040	17.5	24	4.6
25	588425	19.2	25575	14.6
26	312156	18.3	1638	10.7
27	67554	16.0	13797	13.7
28	249165	17.9	28	4.8
29	1466066	20.5	475107	18.9
30	306120	18.2	30	4.9
31	2841150	21.4	31	5.0
32	2002272	20.9	1	0.0
33	2038476	21.0	1023	10.0
34	5656002	22.4	510	9.0
35	18480630	24.1	4095	12.0
36	2237472	21.1	252	8.0
37	49276415	25.6	3233097	21.6
38	9329228	23.2	19418	14.2
39	961272	19.9	4095	12.0
40	19211080	24.2	120	6.9
41	51151354	25.6	41943	15.4
42	109603410	26.7	126	7.0
43	93537212	26.5	5461	12.4
44	192218312	27.5	1364	10.4
45	75864495	26.2	4095	12.0
46	261598274	28.0	4094	12.0
47	811284813	29.6	8388607	23.0
48	30359186?6	31.5	48	5.6
49	9937383652	33.2	2097151	21.0
50	593487780	29.1	51150	15.6
51	36257110?3	31.8	255	8.0
52	20653434880	34.3	3276	11.7
53	40114679273	35.2	3556769739	31.7
54	7551779562	32.8	27594	14.8

The cycle structures of finite cellular automata depend in detail on the boundary conditions chosen. Table 9.3 gives the maximal cycle lengths found for rules (3.1) and (2.4) with shift register boundary conditions. The results differ substantially from those with periodic boundary conditions given in Table 9.2. One notable feature is the presence of length $2^N - 1$ cycles in the linear cellular automaton (2.4) for certain N. These correspond to maximal length linear feedback shift registers, and can be identified by a direct algebraic procedure [4]. Table 9.3

Maximum cycle lengths π_N found for the cellular automata of Eqs. (3.1) (CA30) and (2.4) (CA60) in shift registers of size N (with boundary conditions given by Eq. (2.3)).

	CA30		CA60	
N	π_N	$\log_2 \pi_N$	π_N	$\log_2 \pi_N$
4	5	2.3	15	3.9
5	2	1.0	21	4.4
6	7	2.8	21	4.4
7	4	2.0	127	7.0
8	17	4.1	63	6.0
9	65	6.0	73	6.2
10	6	2.6	889	9.8
11	57	5.8	1533	10.6
12	50	5.6	1085	10.1
13	118	6.9	7905	12.9
14	185	7.5	11811	13.5
15	257	8.0	32767	15.0
16	481	8.9	255	8.0
17	907	9.8	273	8.1
18	1681	10.7	253921	18.0
19	707	9.5	413385	18.7
20	2679	11.4	761763	19.5
21	5630	12.5	5461	12.4
22	1368	10.4	4194303	22.0
23	31241	14.9	2088705	21.0
24	3567	11.8	2097151	21.0
25	60503	15.9	2192337	21.1
26	4752	12.2	22995	14.5
27	46519	15.5	41943035	25.3
28	35569	15.1	17895697	24.1
29	207197	17.7		
30	149899	17.2		
31	482717	18.9		

Other boundary conditions may also be considered. Among them are twisted ones, in which the sites a_1 and a_N are negated in Eq. (2.2). The maximum cycle lengths found with such boundary conditions seem typically shorter than in the purely periodic case.

One may in addition consider boundary conditions in which the boundary site values are fixed, rather than being periodically

identified. Section 6 (particularly Fig. 6.2) gave some examples of results with such boundary conditions. Different cycles are obtained in different cases; all those investigated nevertheless give maximal cycle lengths shorter than those of Table 9.2 found with periodic boundary conditions.

What has been discussed so far are cycles in complete finite cellular automaton configurations. But in obtaining random sequences one samples single sites. The sequences found could potentially have periods which were sub-multiples of the periods for the complete configuration. For permutive rules such as (3.1) (or (2.4)) this cannot, however, occur.

The state transition diagrams summarized in Table 9.1 give the number of complete N-site configurations that can occur at various stages in the evolution of the cellular automaton (3.1). One may also consider the number of single site temporal sequences that can occur. Table 9.4 gives the fraction of the 2^L possible length L temporal sequences that are actually generated from any of the 2^N possible initial states in a size N cellular automaton evolving according to Eq. (3.1) (with periodic boundary conditions). The results are plotted in Fig. 9.3. Whenever $N \gtrsim L+2$, all possible sequences seem to be generated. They appear with roughly equal frequencies.

Previous section | Next section

10. Statistical Properties

The sequences generated by the cellular automaton of Eq. (3.1) may be considered effectively random if no feasible procedure can identify a pattern in them, or allow their behaviour to be predicted. Even though it may not be possible to prove that no such procedure can exist, circumstantial evidence can be accumulated by trying various statistical procedures and finding that they reveal no regularities. The basic approach is to compare statistical results on sequences generated by (3.1) with those calculated for sequences whose elements occur purely according to probabilities.

To establish the validity of (3.1) as a general-purpose random sequence generator, one should apply a variety of statistical procedures, related to various different kinds of calculations. The choice of tests is necessarily as ad hoc as the choice of calculations done. Appendix A lists those used here. (But see also [35].) Some can be considered related to Monte Carlo simulations of physical and other systems. Others to statistical analyses that would be done on data from various kinds of measurements. While quite ad hoc, the tests seem to be sensitive, and reasonably independent.

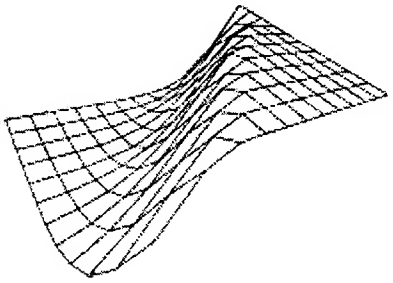
Table 10.1

Fraction of length L temporal sequences generated from all possible seeds by evolution according to Eq. (3.1) in a length N circular register. Results for successive values of N are given in successive columns. The results are plotted in Fig. 9.3.

L	3	4	5	6	7	8	9	10	11	12	13	14	15
Opt 3	0.500	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Opt 4	0.250	0.625	0.875	0.938	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Opt 5	0.125	0.313	0.656	0.844	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Opt 6	0.063	0.156	0.344	0.594	0.906	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Opt 7	0.031	0.078	0.180	0.352	0.609	0.891	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Opt 8	0.016	0.039	0.094	0.188	0.328	0.633	0.949	0.992	1.000	1.000	1.000	1.000	1.000
Opt 9	0.008	0.020	0.047	0.094	0.168	0.361	0.668	0.895	0.996	1.000	1.000	1.000	1.000
Opt10	0.004	0.010	0.023	0.047	0.085	0.195	0.386	0.644	0.917	0.989	1.000	1.000	1.000
Opt11	0.002	0.005	0.012	0.023	0.042	0.102	0.204	0.377	0.666	0.897	0.995	1.000	1.000
Opt12	0.001	0.002	0.006	0.012	0.021	0.052	0.105	0.204	0.387	0.651	0.911	0.995	1.000
Opt13	0.000	0.001	0.003	0.006	0.011	0.026	0.054	0.105	0.209	0.385	0.669	0.913	0.995
Opt14	0.000	0.001	0.001	0.003	0.005	0.013	0.027	0.053	0.109	0.209	0.397	0.671	0.906
Opt15	0.000	0.000	0.001	0.001	0.003	0.007	0.013	0.027	0.055	0.109	0.215	0.399	0.668

Figure 10.1

Fraction of length L sequences obtained by evolution from all possible seeds according to Eq. (3.1) in a size N circular register. The three-dimensional view is from the point $N = L = 20$, with elevation 2.



Enlarge

As an example, consider the ``equidistribution'' or ``frequency'' test. If a sequence of zeroes and ones is to be random, the digits zero and one must occur in it with equal frequency. In general, in fact, all 2^n possible length n blocks of digits must also occur with equal frequency. (The measure entropy of (4.2) is maximal exactly when such equidistribution occurs.) However, in a finite sample of length m, there are expected to be statistical fluctuations, which lead to slightly different numbers of zeroes and ones. (The value of entropy deduced from a finite sample is thus almost always not maximal, even if it would be maximal were the sequence to be continued forever.) As a consequence, one can never definitively conclude by studying a finite sample that the complete sequence is not random. One can however calculate the probability that a truly random sequence would have the properties seen in the finite sample.

To do this (e.g., [36]), one evaluates χ^2, defined in terms of the observed and expected frequencies p_0 and p_c as

$$\chi^2 = \sum_1^\nu (p_0 - p_c)^2 / p_c. \tag{10.1}$$

Here ν gives the number of degrees of freedom, or number of distinct objects whose frequencies are included in the sum. If blocks of length n are studied then $\nu = 2^n$. Now one must find the probability that a value of χ^2 larger than that observed would occur for a random sequence. This ``confidence interval" is obtained immediately from the integral of the χ^2 distribution (e.g., [36]).

If the confidence interval is very close to zero or one, then the observed χ^2 is unlikely to be produced from a random sequence, and one may infer that the observed sequence is not random. Of course, if say a total of k tests are done, it is to be expected that the confidence interval for at least one of them will be less than $1/k$. Evidence for nonrandomness in a sequence must come from an excess of confidence interval values close to zero or one, over and above the number expected for a uniform distribution.

Table 10.1 gives results from the statistical tests described in Appendix A for sequences generated by the cellular automaton (3.1) in a finite circular register. Except when the sample sequence is comparable in length to the period of the system, as given by Table 9.2, no significant deviations from randomness are found. Table 10.2

Results of the statistical tests described in Appendix A for sequences of length L ($k = 1024$) generated by the cellular automaton of Eq. (3.1) (rule number 30) in circular registers of length N. In each case, the seed used consists of a single nonzero site. The numbers given are the probabilities (confidence intervals) for statistical averages of truly random sequences to exceed those of the sequences analysed. The numbers should be uniformly distributed between 0 and 1 if the sequences analysed are indeed truly random. Results below 0.05 and above 0.95 are shown in bold type. Accumulations close to 0 or 1 suggest deviations from randomness. Such accumulations are seen in this case only when the period of the cellular automaton is comparable to the length of the sequence sampled. (The statistical test programs used here were written in C by Don Mitchell.)

	CA30 $N = 17$ $L = 8k$	CA30 $N = 17$ $L = 64k$	CA30 $N = 23$ $L = 64k$	CA30 $N = 29$ $L = 64k$	CA30 $N = 37$ $L = 64k$	CA30 $N = 49$ $L = 64k$
A	0.0039	1.0000	0.0456	0.7375	0.3852	0.8003
B	0.0171	0.9944	0.3391	0.4888	0.1010	0.1494
C	0.4164	0.4783	0.7256	0.4847	0.4083	0.9407
D	0.3227	0.9998	0.1506	0.1434	0.1678	0.6074
E	0.4576	0.4484	0.6790	0.8492	0.5414	0.7991
F	0.4306	0.8644	0.8751	0.5590	0.6681	0.6606
G	0.2942	0.9944	0.1232	0.7359	0.4448	0.6961

Table 10.3

Results of statistical tests for sequences generated by various procedures. CA60 is the linear cellular automaton rule of Eq. (2.4), in a size N circular register. LFSR is a linear feedback shift register of length N with period $2^N - 1$. For $N = 17$ the shift register taps are at positions 14 and 17; for $N = 29$ they are at positions 27 and 29. For CA60 and LFSR seeds consisting of a single nonzero site were used. LCG is the linear congruential generator $x' = (1103515245x + 12345) \bmod 2^{31}$ (used, for example, in many implementations of the UNIX operating system). The seed $x = 1$ was used. The behaviour of CA60, LFSR and LCG are illustrated in Fig. 11.1. $\sqrt{2}$, e and π are the binary digit sequences of the square root of two, the exponential constant, and pi, respectively. (These digit sequences were obtained by R. W. Gosper using a Symbolics 3600 LISP machine.)

	CA60 $N = 29$ $L = 64k$	LFSR $N = 17$ $L = 64k$	LFSR $N = 29$ $L = 64k$	LCG $N = 32$ $L = 64k$	$\sqrt{2}$ $L = 51906k$	e $L = 9501k$	π $L = 26755k$
A	1.0000	0.0390	0.9998	0.0167	0.6255	0.5505	0.1441
B	1.0000	0.9773	0.4378	0.0841	0.0801	0.4556	0.9525
C	1.0000	0.2654	1.0000	0.1676	0.0582	0.8615	0.2799
D	1.0000	0.8797	0.8400	0.8322	0.8553	0.7605	0.9986
E	0.9256	1.0000	0.9435	0.5850	0.6363	0.6890	0.0049
F	0.9998	1.0000	0.9674	0.9248	0.8499	0.7031	0.1297
G	1.0000	0.9790	0.3476	0.3137	0.8465	0.4086	0.5473

Table 10.2 gives statistical results for sequences generated by other procedures. Those obtained from linear feedback shift registers, while provably random in some respects (e.g., [4]), are revealed as significantly nonrandom by several of the tests used here. Many sequences obtained from linear congruential generators are also found to be significantly nonrandom with respect to these tests. No regularities are detected in the digit sequence of $\sqrt{2}$ (and other surds tried) (cf. [37]). There is, however, some possible evidence for nonrandomness in the digit sequences of e and π (cf. [38]). (This will be explored elsewhere.)

Table 10.3 gives statistical results for temporal sequences in the pattern of Figure 6.1 obtained by evolution according to Eq. (3.1) from a single nonzero initial site on an infinite lattice. Once again, no significant deviations from randomness are seen.

If deviations from randomness were detected by some statistical procedure, then this procedure could be used to make statistical predictions about the sequence. In addition, it could be used to obtain a compressed representation for the sequence, and would thus demonstrate that the sequence did not have maximal information content. The fact that deviations from randomness have not been found by any of the statistical procedures considered lends strong support to the belief that sequences produced by Eq. (3.1) with large N are indeed random for practical purposes.

Table 10.4

Results of statistical tests for vertical sequences at position i in the pattern of Fig. 6.1 generated by evolution according to Eq. (3.1) from a single nonzero initial site on an infinite lattice. Leading zeroes in each sequence were truncated. (The sequences were obtained by Jim Salem using a prototype Connection Machine computer.)

	$i=0$ $L=8k$	$i=0$ $L=64k$	$i=0$ $L=512k$	$i=1$ $L=512k$	$i=-1$ $L=512k$	$i=32$ $L=512k$	$i=-32$ $L=512k$
A	0.1536	0.2234	0.6453	0.8629	0.8630	0.8733	0.2677
B	0.5996	0.0637	0.4891	0.7639	0.8343	0.2525	0.1751
C	0.6448	0.6538	0.5443	0.5887	0.4000	0.8271	0.8815
D	0.5921	0.2643	0.0051	0.0105	0.7030	0.4550	0.7832
E	0.1358	0.1348	0.6631	0.8430	0.7498	0.1264	0.8353
F	0.2622	0.1957	0.9385	0.4324	0.9009	0.4736	0.8022
G	0.4542	0.8773	0.6658	0.1080	0.7169	0.7744	0.2364

Previous section | Next section

11. Practical Implementation

The simplicity and intrinsic parallelism of the cellular automaton rule (3.1) makes possible efficient implementation on many kinds of computers.

On a serial-processing computer, each site could be updated in turn according to (3.1). But in practice, site values can be represented by single bits in say a 32-bit word, and updated in parallel using standard word-wise Boolean operations. (Additional bit-wise operations are often needed for boundary conditions.)

On a synchronous parallel-processing computer, different sites or groups of sites in the cellular automaton can be assigned to different processors. They can then be updated independently (though synchronously), using the same instructions, and with only local communications.

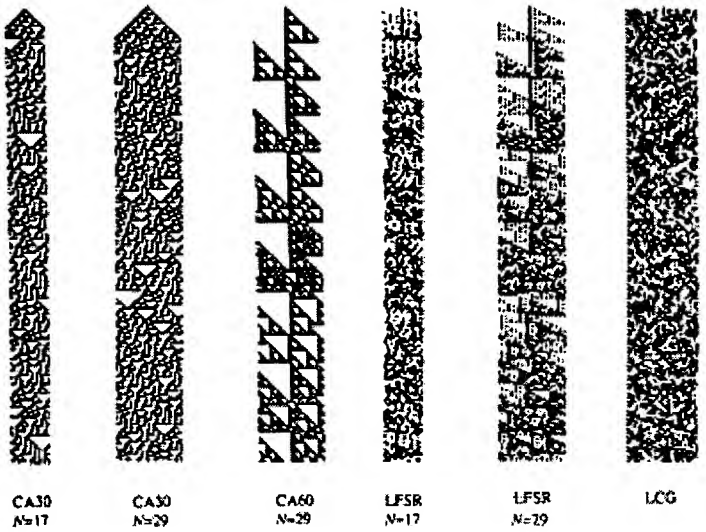
Very efficient hardware implementations of (3.1) should also be possible. For short registers, explicit circuitry can be included for each site. And for long registers, a pipelined approach analogous to a feedback shift register can be used (cf. [39]).

The evidence presented above suggests that the cellular automaton of Eq. (3.1) can serve as a practical random sequence generator. The most appropriate detailed choices of parameters depend on the application intended. The most obvious constraint is one of cycle length. To obtain a cycle length larger than $2^{32} \simeq 4 \times 10^9$, Table 9.2 shows that a circular register of length $N = 49$ can be used. Cycle lengths tend to increase with N, but Table 9.2 shows some irregularities. Thus it is not clear, for example, how large N need be to obtain a cycle length larger than $2^{64} \simeq 10^{19}$. But based on Eq. (9.1), a value $N = 127$ should certainly suffice.

Random sequences can be obtained by sampling the sequence of values of a particular site in a register updated according to Eq. (3.1). The theoretical and statistical studies described above support the contention that such sequences show no regularities. For some critical applications, it may be best however, to sample site values only say on alternate time steps. While this method generates a sequence more slowly, it should foil prediction procedures along the lines discussed in Section 7.

Figure 11.1

Patterns obtained by various procedures in registers of size N. CA30 stands for the cellular automaton of Eq. (3.1), with periodic boundary conditions. CA60 is the linear cellular automaton of Eq. (2.4), again with periodic boundary conditions. LFSR is a linear feedback shift register with size N and period $2^N - 1$. For $N = 17$ the taps are at positions 14 and 17; for $N = 29$, they are at positions 27 and 29. LCG is a linear congruential sequence generator, operating on the 32-bit integers whose binary digit sequences are given. The seed in all cases consists of a single nonzero bit in the center of the register. Statistical properties of the sequences produced are given in Tables 10.1 and 10.2.



Enlarge

Sequences could potentially be obtained more quickly by extracting the values of several sites in the register at each time step. But Eq. (4.6) implies that some statistical correlations must exist between these values. The correlations are probably minimized if the sites sampled are equally spaced around the register. Nevertheless, in some applications where only a low degree of randomness is needed, it may even be satisfactory to use all site values in the register. (An example appears to be approximation of partial differential equations, where randomness can be used to emulate additional low-order digits.)

The random sequences obtained from Eq. (3.1) have an equal fraction of 0 and 1. Many applications, however, involve random binary choices with unequal probabilities. There is nevertheless a simple algorithm [40] to obtain digits with arbitrary probabilities. First write the probability p for outcome 1 as a binary number. Then generate a random binary sequence s with a length equal to

this number. The output is obtained by an iterative procedure. Begin with a ``current result'' of 1. Then, starting from the least significant digit in p, successively find a new result by combining the old result with the corresponding digit of s, using a function AND or OR, depending on whether the digit in p is 0 or 1, respectively. The final result thus obtained is equal to 1 with probability exactly p.

Configurations in two length N registers with slightly different seeds should become progressively less correlated under the action (3.1) as a result of the instability discussed in Section 5. The characteristic time for this process is governed by Eqs. (5.1) and (5.2), and should be $\simeq 0.8N$. Thus, if several sequences are to be generated with seeds that differ only slightly (obtained for example from addresses of computer elements), then (3.1) should be applied at least $O(N)$ times to the seeds before beginning to extract random sequences.

One may compare the scheme for random sequence generation described here with the linear methods now in common use (e.g., [1]). Figure 11.1 shows patterns produced by these various schemes. The primary feature of linear schemes is that they can be analysed by algebraic methods. As a consequence, certain randomness properties can be proved for the sequences they generate, and cases that give long cycles can be identified. But the simplicity in structure which underlies this analysis also limits the degree of randomness that such schemes can produce. The nonlinear scheme described here is not readily amenable to complete analysis, and no significant limits on the degree of randomness it yields are known. But on the other hand, no conventional mathematical proofs for particular randomness properties can be given, and it must be investigated by largely empirical methods.

Previous section | Next section

12. Alternative Schemes

The cellular automaton of Eq. (3.1) is one of the simplest that seems good for random sequence generation. But other cellular automata may also be considered, and some potentially have certain advantages.

Among $k=2$, $r=1$ cellular automata, Eq. (3.2) is the only other serious contender. No direct equivalence between this rule and that of Eq. (3.1) is known, but their properties are very similar. Equation (3.2) gives however [45]

(12.1) $$\lambda_\Sigma = (0.1724 \pm 0.0004),$$

slightly smaller than the corresponding result (5.2) for Eq. (3.1). In addition, it gives a slightly smaller invariant entropy h_μ. It seems to have no advantages over (3.1).

Cellular automata with $k > 2$ or $R > 3$ may also be studied. (Here R is defined as the total number of sites in the neighbourhood for the rule.) Any class 3 (chaotic) cellular automaton rule can be considered a candidate random sequence generator. Autoplectic rules which produce complex patterns even from simple initial conditions are probably best. Some of these rules have larger Lyapunov exponents and invariant entropies than Eq. (3.1), but they are also more difficult to compute. In addition, many rules that seem to produce chaotic overall patterns nevertheless yield sequences that show definite regularities, resulting, for example, in non-maximal temporal entropies. Permutive chaotic rules avoid such problems, but are very similar in character to the rule of Eq. (3.1), and so potentially share any of its possible deficiencies. Table 12.1

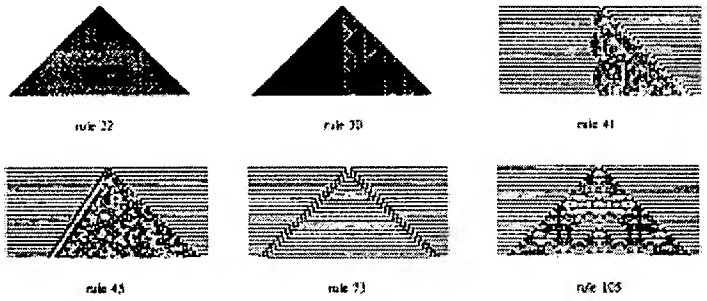
Bijective cellular automata rules with k possible values for each site and depending on strictly R previous site values. The rules given are ``totally quiescent," so that $\phi(a, a, \ldots, a) = a$ for all a. The rules are specified by giving the values of ϕ as digits in a binary number indexed by a number formed from the arguments of ϕ. The binary number is then stated in base 32, with letters of the alphabet representing successive digits greater than 9. Leading zeroes are not truncated. Long specifications correspond to rules with larger values of R.

ϕ	ϕ^{-1}
$k = 2,\ R = 4$	
1kng	1kng
1s5k	1s5k
1hmc	1hmc
1j4s	1j4s
$k = 2,\ R = 5$	
3nh1vc0	3nh1vc0
3ug5vc0	3ug5vc0
39gtvc0	f20nv1jogtvc0

One possibility is to consider bijective cellular automaton rules, which are invertible, so that each configuration has both a unique successor in time, and a unique predecessor. The state transition diagrams for such cellular automata in finite regions with periodic boundary conditions can contain only cycles, and no transients. But only a very small fraction of all cellular automaton rules are bijective, and very few of those that are exhibit chaotic behaviour. Table 12.1 gives some non-trivial bijective cellular automaton rules with $k = 2$ and $R \leq 5$ (cf. [41]). None of those with $R \leq 4$ are chaotic.

Figure 12.1

Patterns generated by various bijective (reversible) $k = 2,\ r = 1$ cellular automata with rules of the form (12.2).



Enlarge

With larger effective k, it is nevertheless possible to construct chaotic bijective rules explicitly. One method [42] yields cellular automaton rules that are most easily stated in terms of dependence on second-to-last as well as immediately preceding site values:

$$a_i^{(t)} = \phi(a_{i-r}^{(t-1)}, \cdots, a_{i+r}^{(t-1)}) \text{ XOR } a_i^{(t-2)} \qquad (12.2)$$

Such rules may be stated in the standard form (2.1) by considering sites with k^2 possible values. Some examples of patterns generated by rules of the form (12.2) are shown in Fig. 12.1. The rules are bijective, so that all states lie on cycles. However, there are often many distinct cycles, each quite short, making the system unsuitable for random sequence generation.

Previous section | Next section

13. Discussion

This paper has used methods from several disciplines to study the behaviour of the nonlinear cellular automaton of Eq. (3.1). Despite the simplicity of its construction, all the approaches taken support the conjecture that its behaviour is so complicated as to seem random for practical purposes. It is remarkable that such a simple system can give rise to such complexity. But it is in keeping with the observation that mathematical systems with few axioms, or computers with few intrinsic instructions, can lead to essentially arbitrary complexity. And it seems likely that the mathematical mechanisms at work are also responsible for much of the randomness and chaos seen in nature.

The simplicity of Eq. (3.1) makes it amenable to highly efficient practical implementation. And the analyses carried out here suggest that the sequences it produces have a high degree of randomness. In fact, if any regularity could be found in these sequences, it would probably have substantial consequences for studies of many complex and seemingly random phenomena.

Previous section | Next section

Appendix A: Statistical Procedures

This Appendix describes the statistical randomness testing procedures used in Section 10. The procedures are mostly taken from [1], although their numbering has been changed slightly. The basic method in each case is to compare an observed distribution with that calculated for a purely probabilistic sequence.

The sequences studied consist of strings of binary bits. In many of the tests, these bits are grouped into blocks: either length 8 (non-overlapping) bytes, or length 4 (non-overlapping) nybbles. The possible bit sequences in these blocks can be represented by integer ``values" between 0 and 255 or 16, respectively.

A. Block Frequency Distribution.

Each of the 2^n possible n-blocks should occur with equal frequency. ($n = 8$ is used.)

B. Gap Length Distribution.

The lengths of runs of n-blocks whose values are all greater than q_2 or less than q_1 should follow a binomial distribution. ($n = 8$, $q_1 = 100$, $q_2 = 200$ are used; runs longer than 16 blocks are lumped together.)

C. Distinct Blocks Distribution.

The frequencies with which p out of q successive m-blocks are distinct should follow a definite distribution. ($m = 4$, $q = 4$ are used.)

D. Block Accumulation Distribution.

The number of successive n-blocks necessary for all possible m-blocks to appear in order as their first m elements should follow a definite distribution. ($n = 8$, $m = 3$ are used; numbers greater than 40 are lumped together.)

E. Permutation Frequency Distribution.

The values of q successive n-blocks should occur in all $q!$ possible orderings with equal frequency. ($n=8$, $q=5$ are used.)

F. Monotone Sequence Length Distribution.

The lengths of sequences in which successive n-blocks have monotonically increasing values should follow a definite distribution. ($n=8$ is used; lengths greater than 6 are lumped together; elements immediately following each run are discarded to make successive runs statistically independent.)

G. Maxima Distribution.

The maximum values of n-blocks in sequences of qn-blocks should follow a power law distribution. ($n=8$, $q=8$ are used.)

Previous section | Next section

Acknowledgments

Many people have contributed in various ways to the material presented here. For specific suggestions I thank: Persi Diaconis, Carl Feynman, Richard Feynman, Shafi Goldwasser, Peter Grassberger, Erica Jen, and John Milnor.

For discussions I thank: Lenore Blum, Manuel Blum, Whit Diffie, Rolf Fiebrich, Danny Hillis, Doug Lind, Silvio Micali, Marvin Minsky, Andrew Odlyzko, Steve Omohundro, Norman Packard, and Jim Reeds.

For help with computational matters I thank: Keira Bromberg, Bill Gosper, Don Mitchell, Bruce Nemnich, Holly Peck, Jim Salem, Tsutomu Shimomura, Steve Strassmann, and Don Webber.

The computer mathematics system SMP [43] was used for some of the calculations. I thank the Science Office of Sun Microsystems for the loan of a SUN workstation on which most of the graphics and many of the calculations were done. And finally I thank Thinking Machines Corporation for the use of a prototype Connection Machine computer [44], without which much more about the cellular automaton of Eq. (3.1) would still be unknown.

Note added in proof. Eq. (3.1) can also be used to generate efficiently a key sequence for stream encryption [46].

Previous section | Next section

References

[1] D. Knuth, ``Seminumerical Algorithms," Addison-Wesley, Reading, Mass., 1981.

[2] A. Shamir, ``On the generation of cryptographically strong pseudorandom sequences," Lecture Notes in Computer Science Vol. 62, p. 544, Springer-Verlag, New York/Berlin, 1981; S. Goldwasser and S. Micali, Probabilistic encryption, *J. Comput. System Sci.* **28** (1984), 270; M. Blum and S. Micali, How to generate cryptographically strong sequences of pseudorandom bits, *SIAM J. Comput.* **13** (1984), 850; A. Yao, Theory and applications of trapdoor functions, in ``Proc. 23rd IEEE Symp. on Foundations of Computer Science," 1982.

[3] G. Chaitin, On the length of programs for computing finite binary sequences, I, II, *J. Assoc. Comput. Mach.* **13** (1966), 547; **16** (1969), 145, Randomness and mathematical proof, *Sci. Amer.* **232**, No. 5 (1975), 47; A. N. Kolmogorov, Three approaches to the concept of ``the amount of information," *Problems Inform. Transmission* **1**, 1 (1965); R. Solomonoff, A formal theory of inductive inference, *Inform. Control* **7** (1964), 1; P. Martin-Lof, The definition of random sequences, *Inform. Control* **9** (1966), 602; L. Levin, On the notion of a random sequence, *Soviet Math. Dokl.* **14** (1973), 1413.

[4] S. W. Golomb, ``Shift Register Sequences," Holden-Day, San Francisco, 1967.

[5] M. Garey and D. Johnson, ``Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman, San Francisco, 1979.

[6] L. Blum, M. Blum, and M. Shub, Comparison of two pseudorandom number generators, *in* ``Advances in Cryptology: Proc. of CRYPTO-82" (D. Chaum, R. Rivest, and A. T. Sherman, Eds.), Plenum, New York, 1983.

[7] W. Alexi, B. Chor, O. Goldreich, and C. Schnorr, RSA/Rabin bits are $\frac{1}{2} + 1/\text{poly}(\log N)$ secure, *in* ``Proc. Found. Comput. Sci.," (1984); U. Vazirani and V. Vazirani, Efficient and secure pseudorandom number generation, *in* ``Proc. Found. Comput. Sci.," (1984).

[8] L. Kuipers and H. Niederreiter, ``Uniform Distribution of Sequences,'' Wiley, New York, 1974.

[9] J. Lagarias, The $3x+1$ problem and its generalizations, *Amer. Math. Monthly* **92** (1985), 3.

[10] K. Mahler, An unsolved problem on the powers of $\frac{3}{2}$, *Proc. Austral. Math. Soc.* **8** (1968), 313; G. Choquet, Repartition des nombres $k(\frac{3}{2})^n$; mesures et ensembles associes, *C. R. Acad. Sci. Paris A* **290** (1980), 575.

[11] S. Wolfram, Origins of randomness in physical systems, *Phys. Rev. Lett.* **55** (1985), 449.

[12] S. Wolfram, Cellular automata as models of complexity, *Nature* **311** (1984), 419.

[13] D. Farmer, T. Toffoli, and S. Wolfram, (Eds.), Cellular automata, *Physica D* **10** Nos. 1, 2, (1984).

[14] S. Wolfram, Cellular automata and condensed matter physics, *in* Proc. NATO Advanced Study Institute on Scaling phenomena in disordered systems, April 1985.

[15] S. Wolfram, Universality and complexity in cellular automata, *Physica D* **10** (1984), 1.

[16] S. Wolfram, Computation theory of cellular automata, *Commun. Math. Phys.* **96** (1984), 15.

[17] S. Wolfram, Statistical mechanics of cellular automata, *Rev. Modern Phys.* **55** (1983), 601.

[18] S. Wolfram, Undecidability and intractability in theoretical physics, *Phys. Rev. Lett.* **54** (1985), 735.

[19] S. Wolfram, Computer software in science and mathematics, *Sci. Amer.* **251** September 1984.

[20] O. Martin, A. Odlyzko, and S. Wolfram, Algebraic properties of cellular automata, *Comm. Math. Phys.* **93** (1984), 219.

[21] J. Milnor, Notes on surjective cellular automaton-maps, Institute for Advanced Study preprint, June 1984.

[22] E. Jen, ``Global Properties of Cellular Automata,'' Los Alamos report LA-UR-85-1218, 1985; *J. Stat. Phys.*, in press.

[23] J. Guckenheimer and P. Holmes, ``Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields,'' Springer-Verlag, New York/Berlin, 1983.

[24] J. Milnor, Entropy of cellular automaton-maps, Institute for Advanced Study preprint, May 1984; Directional entropies of

cellular automaton maps, Institute for Advanced Study preprint, October 1984.

[25] Ya. Sinai, An answer to a question by J. Milnor, *Comment. Math. Helv.* **60** (1985), 173.

[26] N. Packard, Complexity of growing patterns in cellular automata, *in* ``Dynamical systems and cellular automata," (J. Demongeot, E. Goles, and M. Tchuente, Eds.), Academic Press, New York, 1985.

[27] R. Feynman, private communication.

[28] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, ``Logic Minimization Algorithms for VLSI Synthesis," Kluwer, Boston, 1984.

[29] R. Rudell, ``*Espresso* software program," Computer Science Dept., University of California, Berkeley, 1985.

[30] S. Wolfram, Geometry of binomial coefficients, *Amer. Math. Monthly* **91** (1984), 566.

[31] M. Furst, J. Saxe, and M. Sipser, Parity, circuits, and the polynomial-time hierarchy, *Math Systems Theory* **17** (1984), 13.

[32] C. Feynman and R. Feynman, private communication.

[33] M. Minsky, ``Computation: Finite and Infinite Machines," Prentice-Hall, Englewood Cliffs, N.J., 1967.

[34] B. Harris, Probability distributions related to random mappings, *Ann. Math. Statist.* **31** (1960), 1045.

[35] G. Marsaglia, A current view of random number generators, *in* ``Proc. Computer Sci. and Statistics, 16th Sympos. on the Interface," Atlanta, March 1984.

[36] G. W. Snedecor and W. G. Cochran, ``Statistical Methods," Iowa State Univ. Press, Ames, 1967.

[37] W. Beyer, N. Metropolis and J. R. Neergaard, Statistical study of digits of some square roots of integers in various bases, *Math. Comput.* **24** (1970), 455.

[38] S. Wagon, Is $_{\pi}$ normal?, *Math. Intelligencer* **7** (1985), 65.

[39] T. Toffoli, CAM: A high-performance cellular-automaton machine, *Physica D* **10** (1984), 195; K. Steiglitz and R. Morita, A multi-processor cellular automaton chip, *in* ``Proc. 1985 IEEE International Conf. on Acoustics, Speech, and Signal Processing,"

March 1985.

[40] J. Salem, Thinking Machines Corporation report, to be published.

[41] G. Hedlund, Endomorphisms and automorphisms of the shift dynamical system, *Math. Systems Theory* **3** (1969), 320; G. Hedlund, private communication.

[42] N. Margolus, Physics-like models of computation, *Physica D* **10** (1984), 81.

[43] S. Wolfram, ``SMP Reference Manual," Computer Mathematics Group, Inference Corporation, Los Angeles, 1983.

[44] D. Hillis, ``The Connection Machine," MIT Press, Cambridge, Mass., 1985.

[45] P. Grassberger, ``Towards a quantitative theory of self-generated complexity," Wuppertal preprint (1986).

[46] S. Wolfram, Cryptography with cellular automata, *in* ``Proc. CRYPTO 85," August 1985.

Previous section | Next section

Notes

(1) A stricter definition of randomness can be based on the non-existence of simple descriptions [3], rather than merely the difficulty of finding them. None of the sequences discussed here, nor many generally considered random, would qualify according to this definition.

(2) This operation can be performed locally on a base 6 digit sequence, and so can be implemented as a cellular automaton. Given particular finite boundary conditions, it acts like a linear congruential sequence generator (e.g., [1]). But in an infinite region, its behaviour is more complicated, and is related to the so-called $3N+1$ problem [9].

(3) Recent results [45] suggest in fact that $h_\mu \simeq 1 + T^{-(0.6 \pm 0.1)}$, yielding a final value of 1.

Previous section